A red text on a white background

AI-generated content may be incorrect.Adam Wilkie

2025

# CONTENTS

Table of Contents

[CONTENTS 2](#_Toc201871945)

[Definition 4](#_Toc201871946)

[Game Concept and Design 4](#_Toc201871947)

[Outline the scenario, characters, and gameplay mechanics. 4](#_Toc201871948)

[Scenario 4](#_Toc201871949)

[Main Characters: 4](#_Toc201871950)

[Main Gameplay Mechanics: 5](#_Toc201871951)

[Design and describe characters and environment (objects) 5](#_Toc201871952)

[Characters: 5](#_Toc201871953)

[Environments: 5](#_Toc201871954)

[Explain How Characters (Objects) Interact with Each Other and the Environment 6](#_Toc201871955)

[Research on Text-Based RPGs 6](#_Toc201871956)

[Project management tools 7](#_Toc201871957)

[Data dictionary defining key variables and structures 7](#_Toc201871958)

[Gameplay Evidence 25](#_Toc201871959)

[Testing Strategies 29](#_Toc201871960)

[Quality Criteria 29](#_Toc201871961)

[Quality Criteria Checklist 29](#_Toc201871962)

[Test Various Inputs and Outputs: 33](#_Toc201871963)

[Describe use of breakpoints, variable watching, test strings etc 33](#_Toc201871964)

[Evaluation of final game 36](#_Toc201871965)

[Evaluate quality criteria 36](#_Toc201871966)

[Unit Testing 41](#_Toc201871967)

[System Testing 41](#_Toc201871968)

[Box Testing 41](#_Toc201871969)

[The Idea: 46](#_Toc201871970)

[First Issues encountered: 48](#_Toc201871971)

[What I began to add. 51](#_Toc201871972)

[The Restart 54](#_Toc201871973)

[Classes 56](#_Toc201871974)

[Story Continuation 56](#_Toc201871975)

[Areas 57](#_Toc201871976)

[tree 58](#_Toc201871977)

[Field of foe 58](#_Toc201871978)

[CLEAR STATEMENT 59](#_Toc201871979)

[Centering 59](#_Toc201871980)

[Story Continuation 59](#_Toc201871981)

[Back to Field of foe 60](#_Toc201871982)

[TESTERS FEEDBACK 61](#_Toc201871983)

[MASSIVE BUG AT END 63](#_Toc201871984)

[References 64](#_Toc201871985)

[FINAL CODE SUBMISSION 65](#_Toc201871986)

# Definition

Task Definition:

” Translate a classic role-playing adventure game (RPG) from a procedural paradigm to an object-oriented programming (OOP) paradigm using Python:”

OOP:

* Encapsulation
* Abstraction
* Inheritance
* polymorphism

# Game Concept and Design

## Outline the scenario, characters, and gameplay mechanics.

### Scenario

‘The Legend of Helga’ is a search and rescue type of game coded in python and played within the terminal. Within the game offered lore sections are present to explain what happened to Helga and others where it goes straight into fights. The scenario is that a stranger has asked you to help find their friend Helga who was kidnapped by a guy named Trowser.

### Main Characters:

* The Player: A customizable hero, whose actions shape the story. Starts with basic weapons (like an axe).
* Dime: The player’s quirky companion, often offering support, dialogue, and wielding a special sword.
* Cad Bane: Bar guy who gave info
* Helga: A mysterious figure central to the legend. Her identity and fate are uncovered during the game.
* Walking Tree: A powerful boss enemy guarding part of the Forest path.
* Barry, Harry, Larry, Garry: Enemies or minor characters met on various path
* Trowser: Final Boss

### Main Gameplay Mechanics:

* Text-based menu selection (e.g., "Choose Forest, Tavern, or Field of Foe").
* Turn-based battles with health points and attack choices.
* Inventory system with equip able weapons.
* NPC interactions and dialogue.
* ASCII art for environmental storytelling.
* Path-based exploration with branching outcomes

## Design and describe characters and environment (objects)

### Characters:

* Player: Designed with RPG-style customization and stats (health, attack, etc.).
* Dime: Static stats, primarily a narrative-driven companion.
* Cad bane: Guy with information
* Enemies (e.g., Trowser): Each enemy has a unique dialogue, items, attack damage, and HP.
* Helga: Only encountered later in the game, who needed to be saved

### Environments:

* Forest: Home area to a large tree enemy. Forest ASCII art adds depth as it shows what the tree looks like.
* Tavern: A break from combat, helpful dialogue.
* Field of Foe: Intense battle-focused area with numerous fights including end boss.
* Door Area: in field of foe but offers visual ASCII art splitting Left and Right path choice

## Explain How Characters (Objects) Interact with Each Other and the Environment

* Characters interact using text input/output functions.
* Enemy objects are initiated with health and attack values and interact with the player by attacks in turn-based fights.
* Dialogue is displayed using formatted text, mostly cantered.
* Characters can trigger scripted events, such as a reveal scene with Helga after defeating Trowser.
* Inventory affects combat outcomes (e.g., more defence = less damage).
* Environments are interacted with through player choices (e.g., "Go left" or "Go right" at the door ASCII).
* ASCII elements are still but serve to improve immersion and indicate the current location

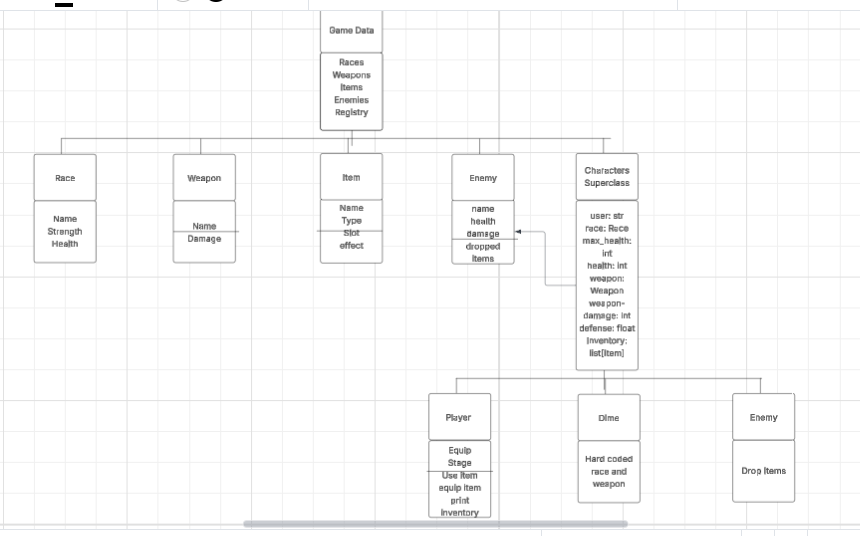
## Research on Text-Based RPGs

Text-based RPGs date back to early computer games like Zork (1980) and Adventure (1976). These games use narrative descriptions, decision-making, and imagination rather than visuals. Key elements include:

* Player choice: Determines direction, outcomes, and replay ability.
* Narrative depth: Storytelling to build emotional engagement writing.
* Combat systems: Often turn-based with stat tracking (HP, items, spells).
* ASCII art: Used historically and stylistically to represent the environment or characters in a visual format.

My game uses all of this with modern Python code, such as interesting storytelling, and an evolving world shaped by player decisions.

# Project management tools

UML Class Diagram

### Data dictionary defining key variables and structures

\*\*\*\*\*NAME TYPE DESCRIPTION IS SAME FOR ALL, IMAGINEW AS ONE BIG TABLE, I SEPERATED IT SO YOU KNEW FROM WHICH FILE\*\*\*\*\*DDD

**Bar.py**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Description | Example |
| Clear\_console | Function | Clears the console (Imported from settings) | clear\_console() |
| columns, rows | Integer | terminal dimensions (from settings.py) | print(line.center(columns)) |
| time | Module | Used for delays with time.sleep() | import time; time.sleep(1) |
| choose\_path | Function | Path selection logic (from choice.py) | choose\_path(player, dime, tree) |
| centered\_input | Function | a user input function aligned to center | name = centered\_input() |
| enter | Integer | Tracks how many times the player enters the bar. If enter == 1, the player has been there before. | Enter == 0 |
| swoosh() | None | Displays cowboy-style ASCII door art centered in the console, pauses, then clears screen. | swoosh() |
| enter\_bar() | player, dime, tree | Manages dialogue and logic for entering the bar, including first-time or repeat visits. | enter\_bar(player, dime, tree) |
| question() | player, dime, tree | Asks the player about Helga’s identity with multiple-choice answers. | question(player, dime, tree) |
| pet() | player, dime, tree | Handles response if player claims Helga is Dime’s friend. | pet(player, dime, tree) |
| friend() | player, dime, tree | Handles response if player claims Helga is "our friend". | friend(player, dime, tree) |
| mother() | player, dime, tree | Handles response if player claims Helga is their mother. | mother(player, dime, tree) |
| grumpy() | player, dime, tree | Handles response if player avoids the question. | grumpy(player, dime, tree) |
| trowser() | player, dime, tree | Describes the character “Trowser” and directs player toward the next location. | trowser(player, dime, tree) |

**Choice.py**

|  |  |  |  |
| --- | --- | --- | --- |
| forest | Function | Begins the forest path (from forest.py) | forest(player, tree) |
| field\_of\_foe | Function | Begins the Field of Foe path (from doors.py) | field\_of\_foe(player, dime, barry, harry, larry, garry, trowser) |
| enter\_bar | Function | Starts the tavern scene (from bar.py) | enter\_bar(player, dime, tree) |
| time | Module | Standard Python module used for delays (e.g., time.sleep()) | Time.sleep(1) |
| player | Object | Player object; passed into all path functions | field\_of\_foe(player, dime, barry, harry, larry, garry, trowser) |
| dime | Object | Companion character; used in some path logic | field\_of\_foe(player, dime, barry, harry, larry, garry, trowser) |
| tree | Object | Big Guy made of wood | forest(player, tree) |
| barry | Object | Bad guy | field\_of\_foe(player, dime, barry, harry, larry, garry, trowser) |
| harry | Object | Bad guy | field\_of\_foe(player, dime, barry, harry, larry, garry, trowser) |
| larry | Object | Bad guy | field\_of\_foe(player, dime, barry, harry, larry, garry, trowser) |
| garry | Object | Bad guy | field\_of\_foe(player, dime, barry, harry, larry, garry, trowser) |
| trowser | Object | Villain from the tavern storyline | field\_of\_foe(player, dime, barry, harry, larry, garry, trowser) |
| centered\_input | Function | Custom input function that centers the prompt text | choice = centered\_input().lower() |

**Forest.py**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Description** | **example** |
| clear\_console | Function | Clears the console screen. Imported from settings.py. | clear\_console() |
| columns, rows | Integer | Terminal dimensions (width and height in characters). From settings.py. | columns, rows = shutil.get\_terminal\_size() |
| time | Module | Python standard module for time-related functions, e.g., sleep(). | import time; time.sleep(1) |
| centered\_input | Function | Presumed custom input function that centers the prompt on the screen. | choice = centered\_input() |
| tree\_picture() | Function | Prints a large ASCII tree illustration. | tree\_picture() |
| tree\_battle(player, tree) | Function | Handles the battle encounter between the player and a walking tree. | tree\_battle(player, tree) |
| forest(player, tree) | Function | Entry point for the forest scene; checks if the tree is dead and starts interaction. | forest(player, tree) |

**Doors.py**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Description** | **eample** |
| clear\_console | Function | Clears the console (from settings.py). | clear\_console() |
| columns, rows | Integer | Terminal dimensions (from settings.py). | columns, rows = shutil.get\_terminal\_size() |
| time | Module | Python standard module for time handling, used for time.sleep(). | import time; time.sleep(2) |
| centered\_input | Function | Custom input function that displays prompts centered on the screen. | choice = centered\_input() |
| choose\_path | Function | Function for rerouting the player’s progression (imported locally in field\_of\_foe). | choose\_path(player, dime, tree) |
| trowser\_picture() | Function | Displays elaborate ASCII art of the final boss (Trowser). | trowser\_picture() |
| door\_picture() | Function | Shows ASCII art of two doors as a narrative/environment cue. | door\_picture() |
| barry\_battle(player, barry) | Function | Manages the first enemy battle with Barry. | barry\_battle(player, barry) |
| harry\_battle(player, harry) | Function | Second battle: against Harry. | harry\_battle(player, harry) |
| larry\_battle(player, larry) | Function | Third enemy battle, against Larry. | larry\_battle(player, larry) |
| garry\_battle(player, garry) | Function | Fourth fight against Garry. | garry\_battle(player, garry) |
| trowser\_battle(player, trowser) | Function | Final boss fight with Trowser. Ends game upon win. | trowser\_battle(player, trowser) |

**Lore.py**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Description** |  |
| clear\_console | Function | Clears the console (from settings.py). | Clear\_copnsole() |
| columns, rows | Integer | Terminal dimensions (from settings.py). | columns, rows = shutil.get\_terminal\_size() |
| os, time, shutil | Modules | Built-in modules used for console manipulation, sleeping, and terminal size. | Import os |
| from start import \* | Various | Imports player and dime-related starting data or constructors. | Start() |
| lore(player, dime) | Function | Handles the introductory lore sequence and establishes the rescue plot. Uses print(), sleep(), and centered\_input() to simulate dialogue. Input: player, dime. Output: None. No branching — story always continues regardless of input ("railroaded"). | print(“im dime, you are”) |
| Scene Intro | Step | Forest scene, mysterious rustling, Dime appears and misidentifies you. | Print(“\*a mysterious figure jumos out the bush\*”) |
| Character Setup | Step | Player introduces self, Dime becomes companion. | Prinmt(‘i am {name}’) |
| Challenge Prompt | Step | Player is asked if they’re "up for the challenge" — always proceeds. | Print(‘are you up for the chal;lenge?’) |
| Main Plot Reveal | Step | Helga has been kidnapped, player is asked again if they'll help. | Print(‘ . . . kidnapped’) |
| Narrative Continuation | Step | Regardless of response, player is assumed to help and story proceeds. | Print(‘soudnds like ay es to me’) |
| Scene Transition | Step | Clears screen, prompts for next location. | Clear\_console() |

**Main.py**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Description** |  |
| start | Function | Initializes the player and companion (from start.py). | Start() |
| lore | Function | Executes the introductory lore scene (from lore.py). | Lore() |
| choose\_path | Function | Handles path/battle navigation for the player (from choice.py). | Choose\_path(player, dime, tree, larry, barry, harry. Garyy, trowser) |

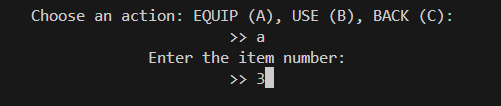
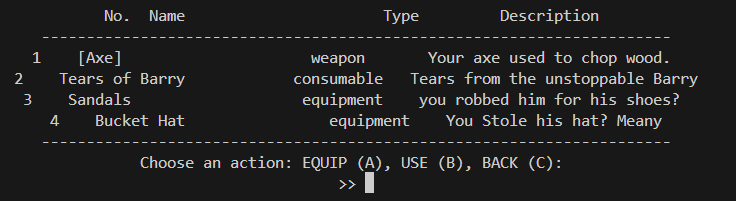
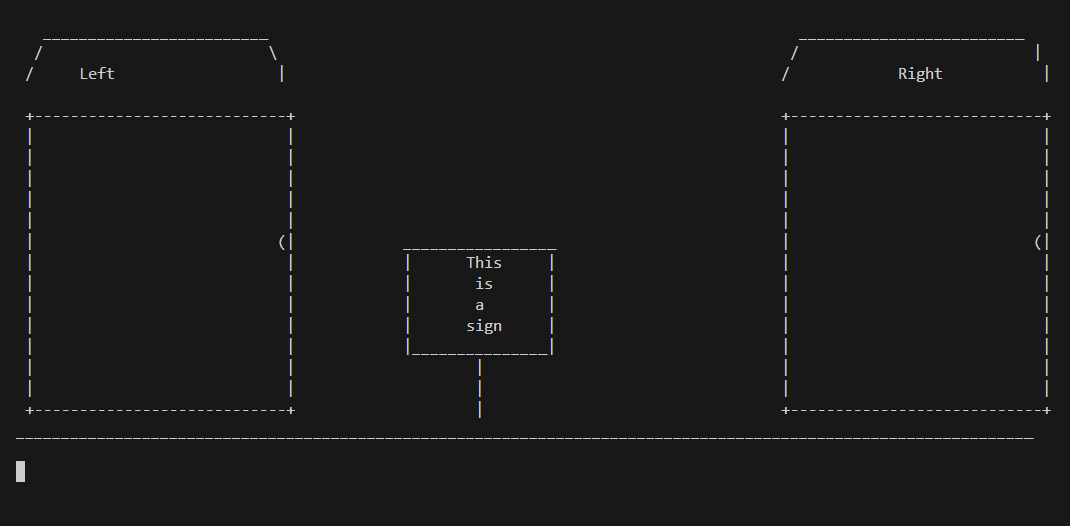
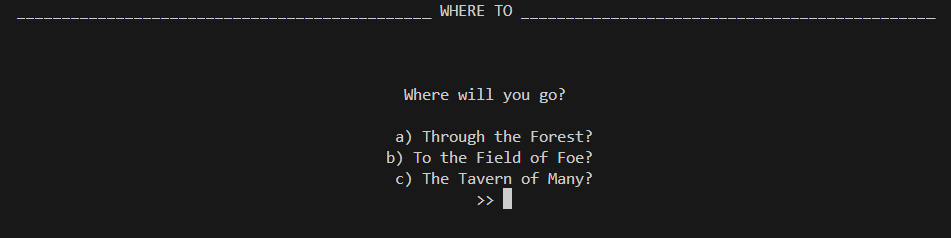
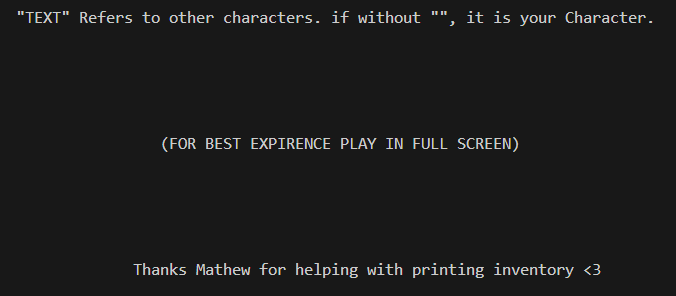
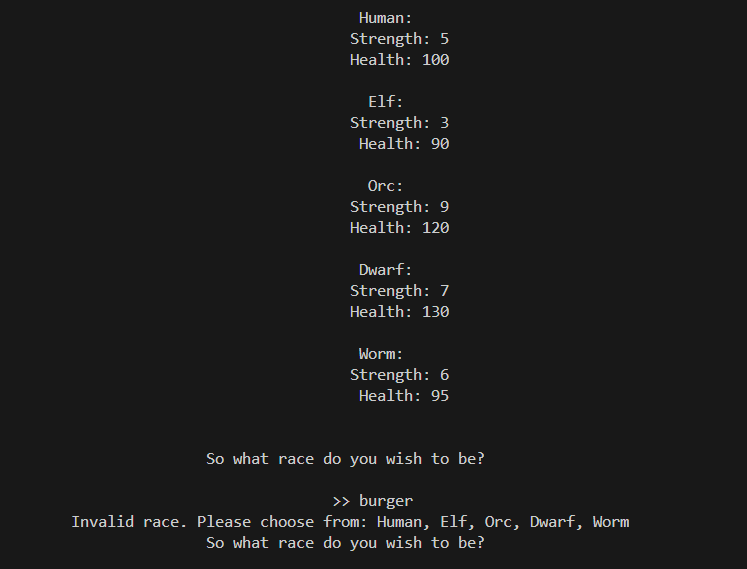
**Settings.py**

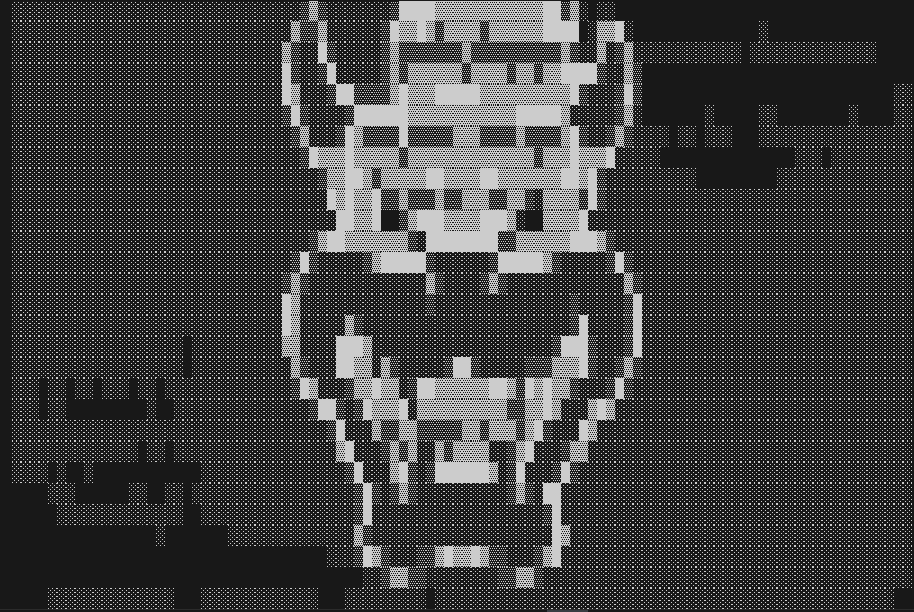
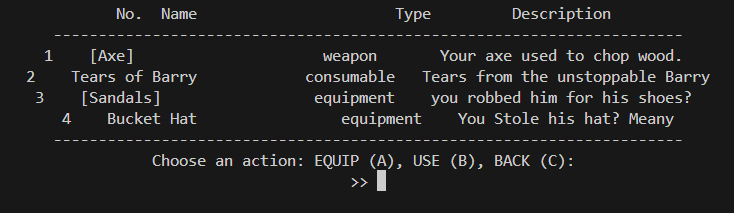
|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Description** |  |
| sys | Module | Used for exiting the game when player dies. | Import sys |
| time | Module | Used for delays and pacing of dialogue. | Import time |
| shutil | Module | Used to get terminal dimensions. | Improt shutil |
| os | Module | Used to clear the console screen. | Import os |
| enter | Int | Placeholder global, currently unused. | Enter == 0 |
| columns, rows | Int | Width and height of terminal; columns used to center output. | columns, rows = shutil.get\_terminal\_size() |
| clear\_console() | Function | Clears the screen for Windows or Unix systems. | os.system('cls' if os.name == 'nt' else 'clear') |
| centered\_input() | Function | Centers an input prompt based on terminal width and returns lowercase. | return input(' ' \* ((columns - len(prompt)) // 2) + prompt).lower() |
| Race | Class | Race template for characters; defines name, strength, health\_bonus. | Class race |
| human | Race | Strength: 5, Health Bonus: 0 — Balanced. | human = Race(  "Human",  strength=5,  health\_bonus=0  ) |
| elf | Race | Strength: 3, Health Bonus: -10 — Fragile. | elf = Race( "Elf", strength=3, health\_bonus=-10 ) |
| orc | Race | Strength: 9, Health Bonus: +20 — Strong. | orc = Race(  "Orc",  strength=9,  health\_bonus=20  ) |
| dwarf | Race | Strength: 7, Health Bonus: +30 — Durable. | dwarf = Race(  "Dwarf",  strength=7,  health\_bonus=30  ) |
| worm | Race | Strength: 6, Health Bonus: -5 — Joke race. | worm = Race(  "Worm",  strength=6,  health\_bonus=-5  ) |
| grug | Race | Strength: 40, Health Bonus: +200 — Secret overpowered. | grug = Race( # SECRET RACES  "Grug",  strength=40,  health\_bonus=200  ) |
| no | Race | Strength: 0, Health Bonus: -95 — Secret underpowered. | no = Race(  "No",  strength=0,  health\_bonus=-95  ) |
| Weapon | Class | Weapon template; defines name, base\_damage. | Class weapon |
| axe | Weapon | Base Damage: 8 — Player's starting weapon. | axe = Weapon(  "Axe",  base\_damage=8  ) |
| dimes\_sword | Weapon | Base Damage: 10 — Dime's default weapon. | dimes\_sword = Weapon(  "Dime's Sword",  base\_damage=10  ) |
| Character | Class | Base character; attributes: user, race, health, weapon, etc. | Class character |
| take\_damage(amount) | Method | Applies damage, factoring in defense; exits if health drops to 0. | def take\_damage(self, amount):  actual\_damage = max(0, int(amount \* self.defense)) # apply defense multiplier  self.health -= actual\_damage  if self.health < 0:  self.health = 0  if self.health == 0:  print(f"{self.user} has died.".center(columns))  if isinstance(self, Player):  print("You died, idiot. Game over.".center(columns))  sys.exit()  return actual\_damage |
| Player | Class | Inherits Character; adds inventory and equipment management. | Class player |
| equipped | Dict | Tracks equipped gear by slot. | self.equipped = {  "head": None,  "feet": None,  "chest": None,  "pants": None,  "shield": None  } |
| equipped\_equipment | Varies | Unused/placeholder. | self.equipped\_equipment = None |
| field\_of\_foe\_stage | Int | Tracks progress through battle zones. | == 0 |
| add\_to\_inventory(item) | Method | Adds item to player inventory. | def add\_to\_inventory(self, item): # adding items to ibv self.inventory.append(item) print(f"{item.name} added to inventory.".center(columns)) |
| use\_item(using) | Method | Uses a consumable item. | def use\_item(self, using): |
| equip\_item(using) | Method | Equips item from inventory. | Def equip\_item(seld, using) |
| print\_inventory() | Method | Displays interactive inventory screen. | def print\_inventory(self): |
| Dime | Class | Companion character; subclass of Character. | Class dime |
| user (Dime) | Str | "Dime". | super().**init**("Dime", 100, human, dimes\_sword) |
| health (Dime) | Int | 100 — Starting HP. | 100 |
| weapon (Dime) | Weapon | Starts with dimes\_sword. | Dimes\_sword |
| Enemy | Class | Inherits Character; adds damage, drop\_items. | Class enemy |
| damage | Int | Base damage enemy deals. | self.damage = damageD |
| drop\_items | List | Items enemy drops on defeat. | self.drop\_items = drop\_items or [] |
| Item | Class | Template for all usable/equippable items. | Class item |
| name | Str | Item name. | self.name = name |
| use\_effect, equip\_effect | Function | What happens on use or equip. | self.use\_effect = use\_effect self.equip\_effect = equip\_effect |
| type | Str | "weapon", "consumable", or "equipment". | axe\_item = Item( |
| description | Str | Description in inventory. | description='Your axe used to chop wood.'  ) |
| slot | Str | Equipment slot (if applicable). | type='equipment' |
| sap\_of\_life | Item | Heals 30 HP — use\_sap\_of\_life() effect. | sap\_of\_life = Item( "Sap of Life", use\_effect=use\_sap\_of\_life, type='consumable', description='Mystical tree juice, capable of healing' ) |
| barrys\_tears | Item | Heals 10 HP — use\_barrys\_tears() effect. | barrys\_tears = Item( "Tears of Barry", use\_effect=use\_barrys\_tears, type='consumable', description='Tears from the unstoppable Barry' ) |
| bark\_shield | Equipment | Slot: shield — Reduces damage (defense = 0.2). | bark\_shield = Item( "Bark Shield", equip\_effect=equip\_bark\_shield, type='equipment', description='Bark Ripped off the tree, deflects damage.', slot='shield' ) |
| bucket\_hat | Equipment | Slot: head — Reduces damage. | bucket\_hat = Item( "Bucket Hat", equip\_effect=equip\_bucket\_hat, type='equipment', description='You Stole his hat? Meany', slot='head' ) |
| sandals | Equipment | Slot: feet — Reduces damage. | sandals = Item( "Sandals", equip\_effect=equip\_sandals, type='equipment', description='you robbed him for his shoes?', slot='feet' ) |
| brown\_stained\_pants | Equipment | Slot: pants — Reduces damage (joke item). | brown\_stained\_pants = Item( "Brown Stained Pants", equip\_effect=equip\_brown\_stained\_pants, type='equipment', description='Some Pants with an awful smell . . . ', slot='pants' ) |
| chopping\_board\_chest\_plate | Equipment | Slot: chest — Reduces damage. | chopping\_board\_chest\_plate = Item( "Chopping Board Chest Plate", equip\_effect=equip\_chopping\_board\_chest\_plate, type='equipment', description='With a chopping board and tape you are protected from danger', slot='chest' ) |
| axe\_item | Item | Default player weapon item. | axe = Weapon(  "Axe",  base\_damage=8  ) |
| dimes\_sword\_item | Item | Item version of Dime’s sword. | dimes\_sword = Weapon(  "Dime's Sword",  base\_damage=10  ) |
| tree | Enemy | Health: 200, Damage: 10, Drops: bark\_shield, sap\_of\_life. | tree = Enemy( "Tree", health=200, damage=10, drop\_items=[bark\_shield, sap\_of\_life] ) |
| barry | Enemy | Health: 15, Damage: 2, Drops: barrys\_tears. | barry = Enemy( "Barry", health=15, damage=2, drop\_items=[barrys\_tears] ) |
| harry | Enemy | Health: 30, Damage: 4, Drops: sandals, bucket\_hat. | harry = Enemy( "Harry", health=30, damage=4, drop\_items=[sandals, bucket\_hat] ) |
| larry | Enemy | Health: 45, Damage: 6, Drops: brown\_stained\_pants. | larry = Enemy( "Larry", health=45, damage=6, drop\_items=[brown\_stained\_pants] ) |
| garry | Enemy | Health: 60, Damage: 10, Drops: chopping\_board\_chest\_plate. | garry = Enemy( "Garry", health=60, damage=10, drop\_items=[chopping\_board\_chest\_plate] ) |
| trowser | Enemy | Health: 200, Damage: 20, Drops: none. | trowser = Enemy(  "Trowser",  health=200,  damage=20,  ) |

**Start.py**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name / Item** | **Type** | **Description** |  |
| \* (from settings) | Various | Imports global functions, variables, classes, races, etc. from settings.py. | From settings impot \* |
| start() | Function | Initializes the game. Displays animated ASCII intro, prompts for player name and race, constructs Player and Dime objects, and returns them. Returns: Tuple (Player, Dime). | def start(): # Start function, called into main |
| clear\_console() | Step | Clears the screen before showing the intro. | clear\_console()# screen clear. |
| ASCII Art Display | Step | Renders a large centered ASCII intro. | some big words im not pasting here but says the legend of helga |
| Name Input | Step | Asks the player what they want to be called. | Name == centered\_input() |
| Final Info Message | Step | Shows player’s chosen stats and controls hint. | print(f'Player created with name: {player.user} and race: {player.race.name} and health: {player.health} '.center(columns)) |
| ascii\_art | Str | Title art string for the game intro. | Ascii\_art |
| ascii\_lines | List[str] | Split lines from ASCII art. | ascii\_lines = ascii\_art.strip('\n').split('\n') |
| vertical\_padding | Int | Padding based on terminal height. | ertical centering vertical\_padding = (rows - len(ascii\_lines)) // 2 # verticles for centering |
| name | Str | Player's chosen name. | name |
| race\_selection | Str | Formatted string of visible race options. | race\_selection ='''  Human:  Strength: 5  Health: 100    Etc:  “”” |
| race\_select | List[str] | List of lines from race\_selection. | Race\_select = race\_selection |
| valid\_races | List[str] | List of all keys in races dictionary (incl. hidden). | valid\_races = list(races.keys()) |
| race\_input | Str | Raw user input for race selection. | Race\_input== centered\_input() |
| chosen\_race | Race | Object reference from races dictionary. | chosen\_race = races[race\_input] |
| player | Player | Player object created from input. | Dime == DIme() |
| dime | Dime | A fixed companion with preset stats. | Player == Player(name, chosen\_race) |

# Gameplay Evidence





# Testing Strategies

### Quality Criteria

The Criteria i used for my code were the following

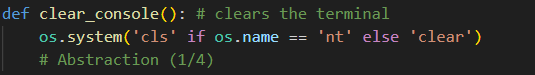
1. OOP to be used
2. Code wasnt ‘too fast’ or ‘too slow’, with sleep times
3. Code didnt look ‘Crazy’ and was well functioned
4. Code was clear and readable
5. Text based input and output
6. Options: like race, name, path etc
7. Consistancy within fights

### Quality Criteria Checklist

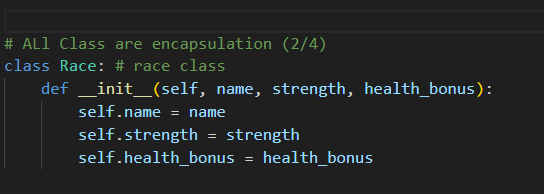
I needed to make sure all were used so i followed down in a checklist whilst coding

OOP to be used:

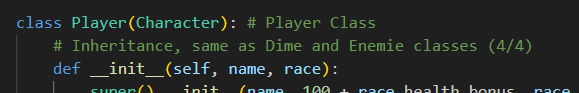
1. Abstraction



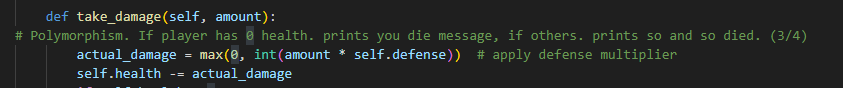
1. Encapsulation



1. Inheritance



1. Polymorphism



Doing this check meant that i have everything i need for the task WITHIN my code.

Code wasnt ‘too fast’ or ‘too slow’, with sleep times:

I didnt want mt text to fly across screen, not giving time to read but then i aslo didnt want it to be so slow that it dragged and i found a great time for this was 3 seconds.

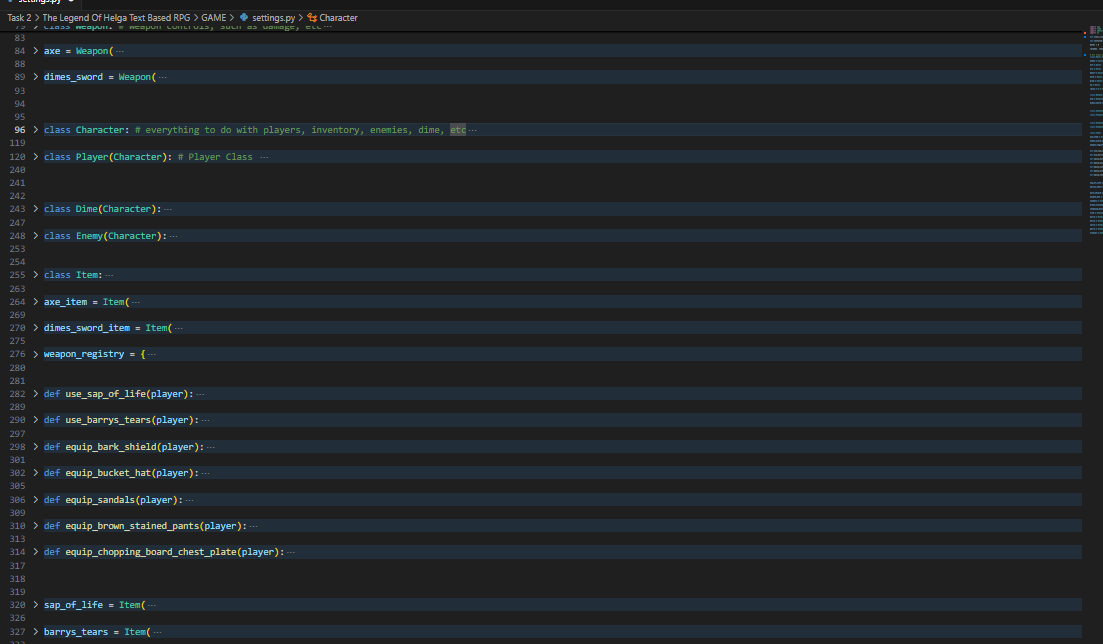
After most prinbt statements i have put

‘Time.sleep(3)’

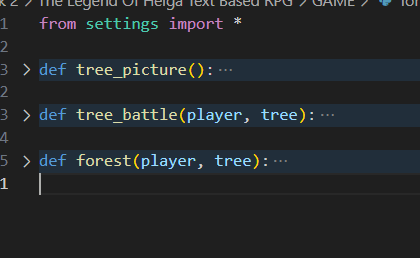
This gives the player 3 seconds to read all the text, for example, in my tavern (bar.py), Where there is lots of print statements, meaning a lot of things for the player to read that added to the plot. Having all this sleep statements made it so the reader wasn't just able to read, but also able to understand what and were happened to Helga

Code didnt look ‘Crazy’ and was well functioned

I Tried seperating my code into well organised classes and functions.

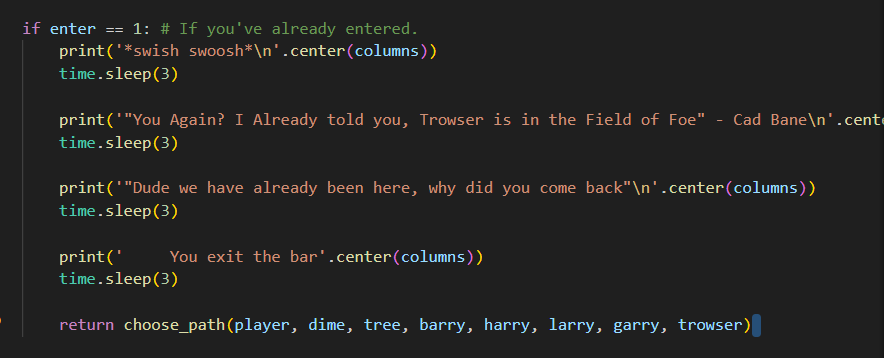
This is Settings.py for example,

For ones text based, i had done like:



Code was clear and readable

I wanted to make sure that someone who does not know how to code could rea dand understand it. With that being said ive tried to keep my code clear, with gaps in betweem to made it look nice.

Look at this as example,

Print

Sleep

Gap

I've done this so it separated each line and their functionality with their partnering sleep to control each part while knowing there is time in-between.

Text based input and output

I have numerous of these. I use them for things such as

Name

Race

Path chocie

Battle scenes

Fight

Go back

Inventory

etc

Options: like race, name, path etc:

I made these all input statements and so the player could choose. It starts with picking the name. Then theres a selection of races amd where you go (which the player chooses) affects path. Within bar.py there multiple option with chat so you can pick your own story

Consistency within fights:

All fights have

Fight

Go Back

Inventory

Besides the fight of tree with the joke option befriend which is just taking damage.

### Test Various Inputs and Outputs:

As mentioned above, I have quite a bit of input and outputs. A way i tested them was running them multiple times and testing different options to see what happens and having them circle back to the same option e.g.,

Race\_picked ‘Pick a race’

If race\_picked

print(test worked)

race\_picked

This way i can test all options and make sure they work and then they will circe back so i can test other inputs.

Describe use of breakpoints, variable watching, test strings etc

A way i tested functions was by doing things like

Def test\_function

print(‘test worked’)

other things my functioned had

Print(‘you stinky ghoul’)

Test\_function()

And then if terminal printed

‘Test Worked’

Then it just confirmed to me that my function work. I had done this with most functions.

Breakpoints:

I had use break points within my code to make sure my functions work and do as i needed. Places where i had put break points include

* In Start() after race input to see waht race is given to th eplayer and make sure it aligns. Another test wihtin this ive done and kept in is a stat checker for the player. After you input all your stats i have it print your name, race and health so the player knows these basic stats about thier character and also so i could make sure it gave out the right things
* In lore() before and after input statements to track the story and make sure it follows correctly after the input statements
* Player.take.damage() to make sure the player takes enough damage.

I did this for much more but this is to just name a few

Variable watching:

I constantly watched variables to make sure they worked correctly like the take damage variable which is enemy damage \* defense, etc.

Test strings:

I did use test strings, a way i had done things like

TEST\_MODE = True

def centered\_input(test):

if TEST\_MODE:

return "orc"

I had used it multiple times. This help me ensure my strings did as i needed. With a project this big, i need to make sure things worked out exactly as needed.

Manual output checking:

I also, manually checked my outputs to make sure they outputted correctly in places such as

‘are you up for challenge’?  
 if no

‘too bad’

‘sounds like a yes’

I had checked that if i do it prints too bad then sounds like a yes but then if i didnt say no it printed sounds like a yes.

System Testing:

I was constantly running my code and checking any new things i added to make sure it worked.

# Evaluation of final game

### Evaluate quality criteria

OOP to be used

Code wasnt ‘too fast’ or ‘too slow’, with sleep times

Code didnt look ‘Crazy’ and was well functioned

Code was clear and readable

Text based input and output

Options: like race, name, path etc

Consistancy within fights

**OOP to be used**

To make sure i had met this criteria (which i had shown earlier) i had pasted a comment in at the first one of each with things like ¼ just so i knew. Doing this created a relief of knowing i had it and could just spit out code without worrying about anything

**Code wasnt ‘too fast’ or ‘too slow’, with sleep times**

I probably have over a hundred time.sleep()’s to ensure that it isnt too fast and ive set the time to 3 seconds to ensure it wasnt too slow and so the user could comprehend and understand it.

**Code didnt look ‘Crazy’ and was well functioned**

I had settings full of classes and functions and then in files that were mostly text (foret, bar, field of foe) i defined all functions at start then called all of them in choice to simply them which i then called in main to simplify again making these documents not crazy hard to read

**Code was clear and readable**

I wanted to make sure you could read and understand my code even if youve never coded before and a way ive made it look easy is by doing

print(‘Hellow World’)

Time.sleep(3)

/\*EMPTY LINE\*/

Repeat

Doing this it im able to show that it prints a line, and waits a second and the that line is over, also, by doing this im able to make sure myself that there is a time statement for most print statements.

**Text based input and output**

With this being a terminal based game, i needed inputs. I had diffrent types of input statements which gave diffrent outputs. One for example is an input with the same output. So like:

‘Are you up for the challenge?’

No matter what you say here it says:

‘Sounds like a yes to me’

\*except for when you say no, it says too bad then sounds like a yes to me\*

Then i had ones that collected and remembered what you had said but said/did the same thing no matter what you did. An example of this is name input. You could make your name anything and it will go to race selection, but your name will be remembered all over the game.

Others were depending on what yous say affects the output such as the race selection. Where you met with an option of races and depending what youy pick will be your race, or path choosing where you are given 3 options on where to go and waht you pick, takes you there. The largest of these i have is probably within bar.py where its asked of the player how you know helga which youre then given 4 options which have their own print statements, which then have thier own option afterwards which then all have the same option telling where the boss fight ios also with an input statement with 3 generational input/outputs.

**Options: like race, name, path etc**

Ever since my planning stage, ive wanted race selections. This was seen in my previous games before deciding on terminal like my pokemon style game which had race swelection. Then my legend of helga game with race selection to finally my terminal style. I always wanted to have this as an option so i felt it had to be there. Same with name cusomization. A game is no fun if you dont get to decide your name.

I wanted to player to have a sense of freedom so i gave them a choice of paths so the game didnt jsut feel like

Name

Race

Go here

keep going

kill boss

This way the player has to find the boss and is met with these 3 options

The forest

Has a semi-boss type like character which is supposed to be hard to kill at the start and for players to come back to if they want

Field of Foe

This was the battle sequence. Fightin a chain of progressively harder characters that seem to go on forever until you finally get to the final boss holding Helga

The Tavern of Many

A Sterotypical western/cowboy style bar with characters as such which explain to you where to go to find helga.

I did it like this so if players went to the forest first. They would see a guy theyd struggle to kill but would want to go back and kill. A place which tells you the path to take. And a long list of battles.

**Consistency within fights**

I wanted consitancy within my fights so players knew what to do without havbing to try do diffrent things for battles

So to keep it simple i gave them 3 options (except for tree)

For the main battles it goes as follows:

Fight

Go Back

Check inventory

This gives the player an option to fight them rather then going straight into it just incase they wanted to leave

Go Back so if they wanted to go fight the tree or didnt want to continue that fight they can go back to the path

Check inventory. This is as all enemys drop an item. Some which lessen the damage taken and others which heal

I have these repeat after every battle

So it may be like

3 OPTIONS

YOU FIGHT

DAMAGE YOU TAKE & DAMAGE ENEMY TAKE

3 OPTIONS

ETC

This was to keep consistency

With Tree i had the same 3 options of fighting, returning or checking inventory but i added a joke option. Befriend tree. I added this to trick players and because the players character isnt expecting it. No matter how much damage protection hey have, they take a flat 20 damage because its unexpected. I thought of adding this so player fall for it – take damage – too scared to take risks. Another difference with the trees fight is after you kill it youre met with 2 options rather then 3. that being Go back and Check inventory. This is as the tree is a standalone enemy and thus there is no battle afterwards.

### Unit Testing

After slightly tweaking the code in any fashion i was running it and trying all options to make sure nothing broke. Its tedious but its the only way to make sure. By doing this i was able to find many issues, major and minor.

For example. Just after getting ready to submit i was testing my game and found out if you went to the bar then the field of foe it crashed as i forgot to update the choose path function call in that file. If i hadnt done prior that could have been a big mistake

### System Testing

I had a lot of System Testers. To name a few, Luke, Matt, Daniel, Krishav, Alex, Etc.

This way i was able to get many inputs for suggestions on the game and how to improve it which was extremely helpful in finalising and adding final touches to the game. Aswell as to discover bugs. As the developer im bias to my code and assumes things work but having multiple people try it im able to find bugs from things i wouldnt have thought of to try.

### Box Testing

**Black-Box testing:**

Black box testing is having someone only try out the game and have them give an analysis on what to fix.

My Black-box tester was Nihar Suthar.

He Gave me many helpful tips and advice. Such as:

Add More time-statements

Slow some timestatements down (they were too fast)

Add some more pictures to hook you back

make a little more interesting

All of this was very helpful and was like giving me another checkbox to tick down such as the OOP checkbox i had. After this testing with Nihar i went straight into fixing these issues and this has benifited my user expirence (UX) greatly.

**Grey-box testing:**

Gray-box testing is having someone look at your game AND the code.

My Gray box tester was my brother Declan Wilkie.

My reason for choosing him is that he no longer lives with me, doesnt go to my school, istn aware of subject and task, etc. But also has advanced knowledge in python and codes for money. So he was completely unbiased but also well trained in the topic.

Suggestions he made was to split my code up and to improve clarity of my text

Originally i had 2 folders, settings and main. He suggested i break up functions and also turn other things into functions and make more files for this. This improvd the clarity and readability of my code and i ended up breaking it up into 8 files.

Settings: Controls main classes and functions

Main: The file you run. Has Start, Lore, and Choose path

Start: Has everything at start of game before text. Name, Race, etc

Lore: Has the bulk of ‘Lore’ at the start before paths

Choice: Has Choose path which has all my path functions called into it

Forest: The tree battle and whatnot

Doors: Has and controls field of foe and battles of field and foe

Bar: Controls the Tavern

This way my code is separated into what they are which makes it easier to read. Additionally, I had a function that was 600 lines long. Doing this broke it up into many functions and the main function calling these ended up becoming 40 lines

**White Box Testing**

White Box testers only look at the code

My white box tester was Mathew Braga.

He had left this for me:

“”

main.py:  
instead from start import \*  
do from start import start()  
  
will make it cleaner, and slightly faster, mainly cleaner  
  
if other files don't need settings, why are they importing it. Be more consistent in imports and what uses what and where.  
  
settings.py  
more intrinsic documentation. What does stuff do and why have you laid it out like this. Why are there random random functions in between two sets of defining different possible items. Settings also isn't a great name, not explanative of what's actually inside.  
  
lore.py  
MORE EXPLANATORY INTRINSIC DOCUMENTATION. Also, only import what you need  
not very obvious which file is next. What do the different file names mean? are they places? And the where should we go first should go at the start of the next file, not the end of this  
  
start.py  
Looks good, not much i would change.  
Why are you talking about chatgpt in your comments. Don't say where you got the code, just say what the code does.  
  
forest.py  
For the tree image, i would recommend it returns the tree image available for printing, not immediately prints it. What if you want the image, but don't want to print it yet, just want to say load everything first  
More comments on your code please  
  
doors.py  
Again, i would recommend returning and formatting the string, not printing it  
mainly same as forest.py  
  
choice.py could just be a function somewhere else, not in a separate file. Doesn't really matter, just lots of importing for not much.  
  
bar.py  
more documentation, for question, instead of saying trowser at end of every individual function, and passing the data through two functions, just say if it was chosen (say trowse=True, then if trowse=True then trowser, or you could say break, to break out of the while loop, then call trowser function, up to you), so that everything is much tidier. Also does question need to be in a function? not entirely sure up to you.

“”

This was incredibly helpful. This gave me things to strive for/try to do.

Somethings i tried to implement but struggled such as the ‘import start()’, for some reason i was unable to get it to work. Others forest, i had it print images etc etc.

JOURNAL:

A red text on a white background

AI-generated content may be incorrect., Picture

**THE JOURNAL**

Adam Wilkie

2025

Contents

# The Idea:

Before beginning my project. I had to find something to base my game on. My original idea was a Pokémon-like game called ‘Critters’, which would have been a pixel style game where you could catch ‘Critters’ to use in battle, and at the start of the game, there would be a character selection screen which could allow you to pick from different races, such as ogre, dwarf, etc. I found a video that looked promising in helping me make this game.

<https://www.youtube.com/watch?v=fo4e3njyGy0>

This was exactly what I was looking for, I could add the character selection screen onto this, and tweak is so I could be my own game, but then I realized that it was the exact same one a previous student had done. I felt as if I had to do something else, so, I decided to do ‘The Legend of Zelda’, I was able to find a video from the same creator that made the Pokémon game, that being “Clear Code”. This Legend of Zelda style game offers things such as monster battles, different weapon selection, magic, stats upgrade, etc.

<https://www.youtube.com/watch?v=QU1pPzEGrqw>

\*\*\*\* ORIGINAL PLAN IDEA NOTES \*\*\*\*

(FROM A NOTEPAD I MADE WHEN WE GOT TASK)

brain storming for software task 2

RPG GAME

Pokémon style;

Critter vs Critter battle game

GAME START:

Play

Race; Human, golem, troll, etc.

Gender; Male, Female

Critter Selection: like Pikachu, Squirtle, etc. but different, fitting towards my game

GAME:

Talking to a NPC who offers rewards for critters you capture. you can keep critters for yourself to use. they can get xp, etc etc

Critters can be combined to 'Evolve' them.

own move sets.

each character has buffs. making some better then others. e.g., one might be able to increase strength of pokemon.

# First Issues encountered:

After installing the GitHub file, I attempted to run the code, but I was faced with many lines of things such as

A black background with red text

AI-generated content may be incorrect., Picture

Saying that the Image directory was wrong, and so I had to go through and fix them all. I had changed all from things like:

Picture 1, Picture

To:

Picture 1, Picture

After doing this, I was able to run the code, which was great but that’s when the next issue struck. I couldn’t move. There should have already been controls,

Movement: WASD

Magic attack: LCTRL

Magic swap: E

Attack: SPACE

Weapon Swap: Q

Stats screen: M

Stat Swap: ARROWS

Stat Upgrade: SPACE

With all these that should be set, I expected them to work but instead I was met with this.

A video game screen with a cartoon character

AI-generated content may be incorrect., Picture

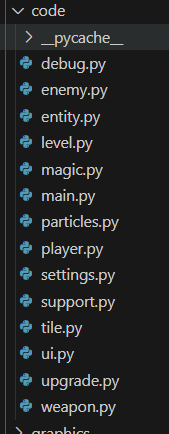
I was unable to move, I was stuck here, nothing was doing anything. But upon looking at the code, I was unable to find that movement wasn’t defined, there was lines there for the movement, except it didn’t work. A screen shot of a computer program

AI-generated content may be incorrect., PictureThis Is the weapon class for example, it has it so where if you direction is let’s say down, you have the down attack animation, except there was nothing like “K\_S”.

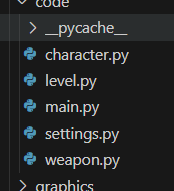
So, I went and added things like

Picture 1, Picture

After a lot of trial and error, I was able to get it working. The next thing I decided to do was clean up the files. Originally it looked like this.



and I had moved a lot of the classes into similar files and so it ended up looking like this.



# What I began to add.

I was still wanting the character selection screen, so I wished to add that but first I thought I needed a title screen. The first thing I did to add the title screen was just make an image for the title screen.



I just made it so any button clicked took you to the next thing I coded, the character selection.

I spent a lot of time on this part. Each character animation was cut up and wasn’t actually a sprite sheet, so I had to spend time manually cutting up the images and ended up with 5 characters.

Sam

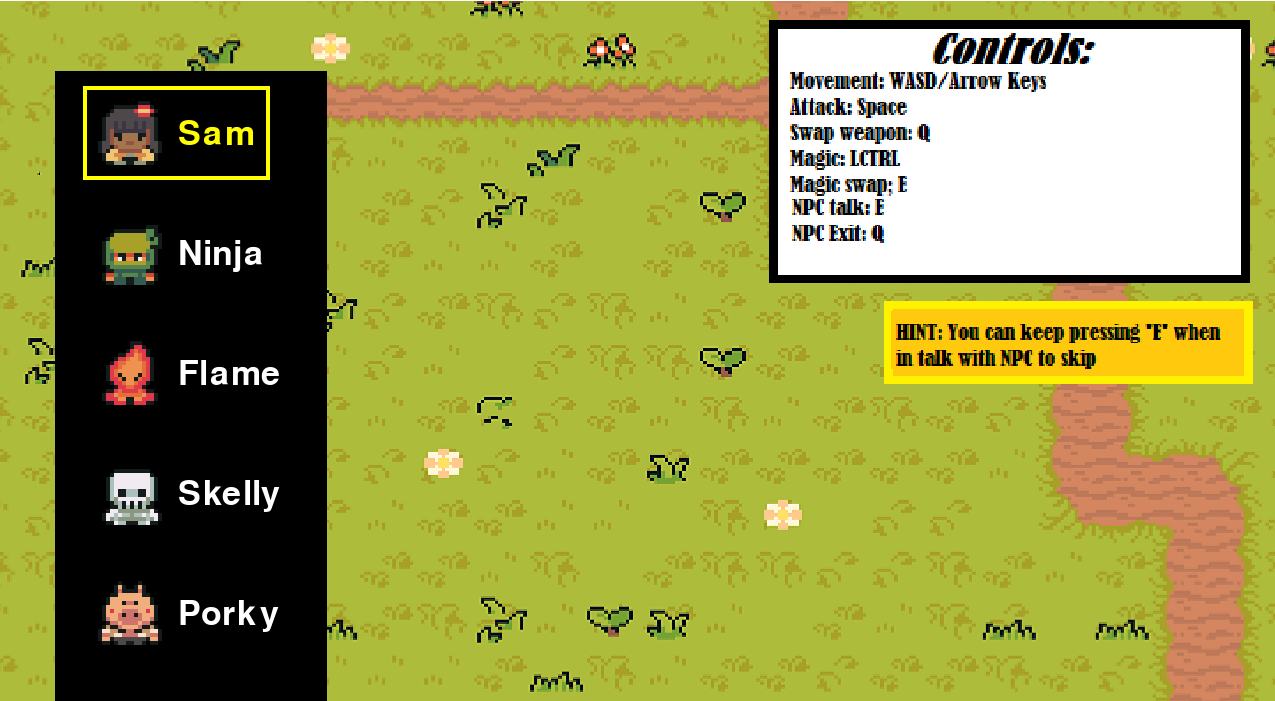
Ninja

Flame

Skelly

Porky

Which looked like this:



After adding this I decided to finally work on the lore. The NPC.

I made a file control file so i knew what everything did and could check oin from tim top time

CHARACTER.PY

Controls NPC

Controls Player

Controls Enemy

Controls Entity

UPGRADE.PY

Controls Weapons

Controls Items

Controls Magic

Controls Upgrades

LEVEL.PY

Controls Camera

Controls Level

Controls UI

Controls Tile

Controls Animation Player

Controls Particles

Controls Debug

Settings.py

Controls Settings

Controls Support

Has weapon data

Has Magic Data

Has Enemy Data

TO DO:

FIX STATS UPGRADE

NPC:

QUEST

"kill 5 bamboo"

Reward

"So and so sword"

I just cut up another image except this time only 1 animation, this made it easy and I was able to have npc as like a talking statue. And after adding some text I came into to the issue of quests. I wanted to add them but I was struggling but when I decided to make a big decision.

# The Restart

I decided to restart and make it terminal based. I feel this was the right decision for continuing with this project as it was becoming too much to make before.

I went into this first making my first print statement which has stuck since.

A black rectangle with white text

AI-generated content may be incorrect., Picture

I didn’t know where I was going with this but I knew I needed lore so I thought id start with this.

Then I thought. What do I want my code to be? What game? Should I go back to my critters? Create something new?

I decided that ill keep my original ideas with race selection but then base the game off my idea for ‘The legend of Helga’.

I had asked a friend, Chatgpt, to make me ASCII art for a title page but it kept giving me

“The Leoend oE helpa”

And I had to fix myself

A black sign with orange letters

AI-generated content may be incorrect., Picture

But then after doing this all I would see in chat is like filename and what not straight into this with my print statement right afterwards, so I started implementing this to “Clear screen”

A screen shot of a computer

AI-generated content may be incorrect., Picture

But, because I had no sleep prints, it went straight into my lore start rather then waiting on the title page.

So I added

A black rectangle with colorful circles and text

AI-generated content may be incorrect., Picture

I now have a working, and good-looking title page section. My next course of action was some classes.

# Classes

I started making classes

I first started with a simple player class and a class for dime.

Since i start with an axe in start of lore. Ill start with an axe and dime can have a wsword cus he swings t me like that

I made a weapon class

I made a race class with all races and stats and a dictrionary of races

# Story Continuation

I went on to continue story, i have a few print statements just saying like im dime youa re?? Helga has been kidnapped etc.

Currently making a choose path function but it keep scrashing game.

I got it working, forgot to put player and waht not in there

Names for area

Field of foe

Bar of mystery

Mystery shack

Creepy dungeon

The forest

Castle

Tavern of many

# Areas

From my ideas i went with

The Forest

Field of Foe

Tavern of many

I will begin coding these withing my choose path function.

I have like a selection screen

“where do you want to go ?

A) The Forest

B) Field of Foe

C) Tavern of many””

If choice == ‘a’

Etc

This seems to work so ill just do for all.

## tree

Im working on forest and ive decided to make the enemy here a big talking tree and you can fight it, go back which i just have as like end cus its i a while loop or befiend

Befriending auto attacks youy for 20 its like a joke area

### Field of foe

I have ascii art of some doors i made . . . i got a lil bored

Uhhmmm i hae same if statements but for left ddoor and right door

I added text before all thiss

I have if left door prints “you see barry’

If rtight door “no enemies’

Ill work on this later

## CLEAR STATEMENT

I found something better then clear \n \* 80, its a long line butr i have in a function called clear\_console()

## Centering

Ive decided to centre all my text. Using stack over flow, youtube, reddit and w3schools i have found a way to do it with text like norman print statements but im struggling with the ascii art so i might just leavce that

I FDOUND A WAY

As in i, i mean chatgpt but

I FOUND A WAY

## Story Continuation

Work on tavern!!

So far my plans for this are a liek sterotypical bar swinging door and like a starwars han solo go to that booth type scene

Some text getting to that up to booth. Got starwars on mind. Helper is called Cad bane. Ill finish this later

### Back to Field of foe

Been a few days, should probobly work on this!

Ive tried continuing the thign and decided to add inventory. I tried alone and then got help by mathew improving it but it wasnt fully working so i had chatgpt help me fix it u[p frpo, what we had.

So i know had an enemy

Barry

I had:

Fight Him

Go back

Inventory

But i want him to drop things

I wanted him to drop his weaponm the stick of anquest but i couldnt figure out how to make it so i can swap weapons so i just mde him dop a consumable which healsplayer as i could figure that out so i have tear sof barry that heal 10 health.

THE TREe I CAN DO FOR TREE!

I went back to the treew to add some iutems for him. I made it so he drop slike tree sap that heals like 30 and then alos like a bark shield which ill do the settings of another day

BACK TO FIELD OF FOE

I triedf working on fights, i made a next enemy Harry. All the same as barry justs lkightly buffer and more strong

## TESTERS FEEDBACK

I had sopme people play my game (NOT AS ANY BOXES I HAVE OTHER PEOPLEF OR THAT) but things recommended where clear code up and breakt up into other files which i did.

I now have 8 files but my code wont run

Imt rying to get battles to play in chouice but they arent working. Will update with fiox

\*\*DONT FORGET TO UPDATE\*\*\*\*\*\*

I forgot to update chocie variable, now its working.

Continue battle scenes, Make an equipment for items in settings and harry drops some armour

GO BACK TO TREE AND DO SHIELD

I made all armour (2 pieces so far) do 0.2 defends (take .2 less damage) but shield is .5

NEXT BATTKIE

LArry, just stronger, drops another armour piece

HAD BLACK BOX, WHITE BOX, GREY BOX TEST AROUND NOW ----- IN DOCUMENTATION

Made garry with more armour and now have 4 pieces, decided finalk boss is going to be a guy named trowser

WENT BACK TO THE TAVRN AND FINISHED IT

Inthe booth i made a conversation happen between cad bane and the player where he told them where trowser could be found and how to find helga,

MADE TROWSER BATTLE

I also got ascii art of bowser face

Finshes trowxer battle.

# MASSIVE BUG AT END

Code and documention was about readfy to submit, i let luke try game out. Found massive bug!!

If you went to the tavren at all, whether that be before you go to battles or you go back from battles into tavern back into battles the code crashed. I spent like 40 misn trying to fix. Forgot to update choose\_path.

# References

For some of these i have not supplied a link becuase i have used them numerous times and would be alot of similar links. using thier name as sumarry, e.g, stack overflow

Chatgpt

Mathew

All people who tested

Declan (helped and explained how to do some functions)

Mr Pap – Resources and assistance

Python Cheat Sheet

HSC Course specification

Stack Overflow

Reddit

W3SCHOOLS

Youtube

Github -- (Original Legend Of Zelda Game)

# FINAL CODE SUBMISSION

MAIN.PY

from settings import tree, barry, harry, larry, garry, trowser

from start import start

from lore import lore

from choice import choose\_path

player, dime = start()

lore(player, dime)

choose\_path(player, dime, tree, barry, harry, larry, garry, trowser)

START.PY

from settings import \*

def start(): # Start function, called into main

clear\_console()# screen clear.

ascii\_art = """

████████╗██╗ ██╗███████╗ ██╗ ███████╗ ██████╗ ███████╗███╗ ██╗██████╗ ██████╗ ███████╗

╚══██╔══╝██║ ██║██╔════╝ ██║ ██╔════╝██╔════╝ ██╔════╝████╗ ██║██╔══██╗ ██╔═══██╗██╔════╝

██║ ███████║█████╗ ██║ █████╗ ██║ ███╗█████╗ ██╔██╗ ██║██║ ██║ ██║ ██║█████╗

██║ ██╔══██║██╔══╝ ██║ ██╔══╝ ██║ ██║██╔══╝ ██║╚██╗██║██║ ██║ ██║ ██║██╔══╝

██║ ██║ ██║███████╗ ███████╗███████╗╚██████╔╝███████╗██║ ╚████║██████╔╝ ╚██████╔╝██╗

╚═╝ ╚═╝ ╚═╝╚══════╝ ╚══════╝╚══════╝ ╚═════╝ ╚══════╝╚═╝ ╚═══╝╚═════╝ ╚═════╝ ╚═╝

██╗ ██╗███████╗██╗ █████╗ █████╗

██║ ██║██╔════╝██║ ██╔═══╝ ██╔══██╗

███████║█████╗ ██║ ██║ ███╗ ███████║

██╔══██║██╔══╝ ██║ ██║ ██║ ██╔══██║

██║ ██║███████╗███████╗╚██████╔╝ ██║ ██║

╚═╝ ╚═╝╚══════╝╚══════╝ ╚═════╝ ╚═╝ ╚═╝

""" # ASCII Art as string

ascii\_lines = ascii\_art.strip('\n').split('\n') # Split lines and calculate vertical centering

vertical\_padding = (rows - len(ascii\_lines)) // 2 # verticles for centering

print('\n' \* vertical\_padding) # Print empty lines to vertically center

for line in ascii\_lines: # Print each line centered horizontally

print(line.center(columns))

time.sleep(5)# Pause before transitioning

clear\_console() # clean screar

print('\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_SELECTION\_SCREEN\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_'.center(columns))# Print selection screen title at top (not vertically centered)

print('\n' \* 3) # Some spacing under the heading

print('What do you wish to be called?: \n'.center(columns) ) # Input a word of sorts, Becomes name.

name = centered\_input()

time.sleep(3)

clear\_console()

print('\n' \* 3)

print('Possible Races:'.center(columns)) # 'Viewable' Races, Theres 2 hidden ones. Grug as Tribute and No for annoying people.

# shown races to user, not including secret (cus yk thier secret)

race\_selection ='''

Human:

Strength: 5

Health: 100

Elf:

Strength: 3

Health: 90

Orc:

Strength: 9

Health: 120

Dwarf:

Strength: 7

Health: 130

Worm:

Strength: 6

Health: 95

'''

race\_select = race\_selection.strip('\n').split('\n')

for line in race\_select:# Print each line centered horizontally

print(line.center(columns))

valid\_races = list(races.keys())# List of races from dictionary in settings

while True: # loop

print('So what race do you wish to be?\n'.center(columns))

race\_input = centered\_input() # Get input from user

race\_input = race\_input.lower() # Make input lowercase

if race\_input in valid\_races: # Check if input is valid

break # Exit the loop if valid

else:

print('Invalid race. Please choose from: Human, Elf, Orc, Dwarf, Worm'.center(columns))

chosen\_race = races[race\_input]

time.sleep(3)

dime = Dime() # so you can do thinks like player.user etc later on

player = Player(name, chosen\_race)

clear\_console()

print(f'Player created with name: {player.user} and race: {player.race.name} and health: {player.health} '.center(columns))

time.sleep(3)

clear\_console() # Hint like screen

print('"TEXT" Refers to other characters. if without "", it is your Character. \n\n\n\n\n'.center(columns))

print('(FOR BEST EXPIRENCE PLAY IN FULL SCREEN)\n\n\n\n\n'.center(columns))

print('Thanks Mathew for helping with printing inventory <3'.center(columns))

time.sleep(5)

clear\_console()

return player, dime

LORE.PY

from settings import \*

import os, time, shutil

from start import \*

def lore(player, dime): # lroe at start of game

columns, \_ = shutil.get\_terminal\_size()

name = player.user # Back story lore ish idk

print('You are in a forest. Chopping wood. When suddenly you hear . . .\n\n\n'.center(columns))

time.sleep(4)

print('\*Bushes rustle in the distance\*\n'.center(columns))

time.sleep(3)

print('\*It gets louder as it approaches you\*\n'.center(columns))

time.sleep(3)

print('\*A figure jump out of the bush, wielding a sword.\*\n'.center(columns))

time.sleep(3) # First Interaction with other characters.

print('"YOU BEAST ILL GET YOU"\n'.center(columns))

time.sleep(3)

print('"Wait . . ."\n'.center(columns))

time.sleep(3)

print('"Youre no monster. . . ."\n'.center(columns))

time.sleep(3)

print('"So uhhh . . . anyways, Im Dime, and you are?"\n'.center(columns))

time.sleep(3.5)

print(f'I am {name}'.center(columns))

time.sleep(4)

clear\_console()

print(f'Partner created with name: {dime.user} and health: {dime.health} and race: {dime.race.name}'.center(columns))#new character

time.sleep(5)

clear\_console()

print('The Story Continues . . . '.center(columns)) # Story Continuation

time.sleep(4)

clear\_console()

print(f'"Thats a lovely name. {name}, I love it!"\n'.center(columns))

time.sleep(3)

print('"Anyways . . . I need your help. And i need it now."\n'.center(columns))

time.sleep(3)

print('"Are you up for the Challenge?"\n'.center(columns))

time.sleep(2)

first\_no = centered\_input()

time.sleep(3) # No Matter whats said, its a yes so storyc an continue

if first\_no.lower() in ['no', 'nah', 'nope', 'nuhuh', 'nuh huh', 'nuh uh', 'nop', 'nup']:

print('Too bad'.center(columns))

time.sleep(2)

print('\n')

print('"Sounds like a Yes to me!"\n'.center(columns))

time.sleep(3)

# Main point of story. To save Helga.

print('"Im getting side tracked here, there are more important measures at stake"\n'.center(columns))

time.sleep(3)

print('"Its . . . "\n'.center(columns))

time.sleep(3)

print('"Its . . My friend Helga, shes been uhh . . . "\n'.center(columns))

time.sleep(3)

print('"Kidnapped."\n'.center(columns))

time.sleep(3)

print('"Are you still able to help me?"\n '.center(columns))

second\_no = centered\_input().lower()

time.sleep(3) # same as before

if first\_no.lower() in ['no', 'nah', 'nope', 'nuhuh', 'nuh huh', 'nuh uh', 'nop', 'nup']:

print('\n')

print('Too bad\n'.center(columns))

time.sleep(2)

print('\n')

print('"Still sounds like a Yes to me!'.center(columns))

time.sleep(6)

clear\_console()

print(f'Where Should we go first {name}?'.center(columns))

time.sleep(4)

CHOICE.PY

from settings import \*

from forest import forest

from doors import field\_of\_foe

from bar import enter\_bar

import time

tree = Enemy( # All Enemy stats repeated here (ik is in settigs but done do like barrry=none)

"Tree",

health=200,

damage=10,

drop\_items=[bark\_shield,

sap\_of\_life]

)

barry = Enemy(

"Barry",

health=15,

damage=2,

drop\_items=[barrys\_tears]

)

harry = Enemy(

"Harry",

health=30,

damage=4,

drop\_items=[sandals, bucket\_hat]

)

larry = Enemy(

"Larry",

health=45,

damage=6,

drop\_items=[brown\_stained\_pants]

)

garry = Enemy(

"Garry",

health=60,

damage=10,

drop\_items=[chopping\_board\_chest\_plate]

)

trowser = Enemy(

"Trowser",

health=200,

damage=20,

)

def choose\_path(player, dime, tree, barry=None, harry=None, larry=None, garry=None, trowser=None, progress\_stage=0): # path options

while True:

clear\_console()

print('\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ WHERE TO \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n\n\n'.center(columns))

print('Where will you go?\n'.center(columns))

print('a) Through the Forest?'.center(columns))

print('b) To the Field of Foe?'.center(columns))

print('c) The Tavern of Many?'.center(columns))

choice = centered\_input().lower()

time.sleep(2)

clear\_console()

if choice == 'a':

clear\_console()

forest(player, tree)

elif choice == 'b':

clear\_console()

field\_of\_foe(player, dime, barry, harry, larry, garry, trowser)

elif choice == 'c':

clear\_console()

enter\_bar(player, dime, tree)

else:

print('Not an Option'.center(columns))

time.sleep(2)

FOREST.PY

from settings import \*

def tree\_picture(): # tree ascii art

clear\_console()

tree\_ascii = '''

.

. ;

. . ;% ;;

, , :;% %;

: ; :;%;' .,

,. %; %; ; %;' ,;

; ;%; %%; , %; ;%; ,%'

%; %;%; , ; %; ;%; ,%;'

;%; %; ;%; % ;%; ,%;'

`%;. ;%; %;' `;%%;.%;'

`:;%. ;%%. %@; %; ;@%;%'

`:%;. :;bd%; %;@%;'

`@%:. :;%. ;@@%;'

`@%. `;@%. ;@@%;

`@%%. `@%% ;@@%;

;@%. :@%% %@@%;

%@bd%%%bd%%:;

((o))((o));

@@%@@%%%::;

(MOUTH :p); . '

%@@@o%;:(.,'

`.. %@@@o%::;

`)@@@o%::;

%@@(o)::;

.%@@@@%::;

;%@@@@%::;.

;%@@@@%%:;;;.

...;%@@@@@%%:;;;;,..

'''

tree\_lines = tree\_ascii.strip('\n').split('\n')

vertical\_padding = (rows - len(tree\_lines)) // 2

print('\n' \* vertical\_padding)

for line in tree\_lines:

print(line.center(columns))

time.sleep(5)

#clear\_console()

def tree\_battle(player, tree): # for the battle that coms up

while True:

clear\_console()

print('You see a Walking Tree. You can either:'.center(columns))

print('a) Fight it'.center(columns))

print('b) Go back'.center(columns))

print('c) Befriend it'.center(columns))

print('d) Check your Inventory'.center(columns))

time.sleep(3)

fight = centered\_input()

if fight == 'a':

clear\_console()

print('You get your weapon ready and engage in battle... against a tree?\n\n'.center(columns))

time.sleep(3)

print(' You attack the Tree and it smacks you back'.center(columns))

player.take\_damage(tree.damage)

tree.health -= player.weapon\_damage

if tree.health < 0:

tree.health = 0

print(f"{player.user}'s health: {player.health}".center(columns))

print(f"{tree.user}'s health: {tree.health}".center(columns))

time.sleep(3)

if tree.health == 0:

print('\*THUD\*. The tree drops.'.center(columns))

print(" It dropped some items.".center(columns))

for item in tree.drop\_items:

player.add\_to\_inventory(item)

while True:

clear\_console()

print('You keep walking through the forest.\n'.center(columns))

print('What do you want to do?\n'.center(columns))

print('a) Go back'.center(columns))

print('b) Check inventory'.center(columns))

choice = centered\_input()

if choice in ['a', 'go back']:

print("You head back the way you came...\n".center(columns))

return # Go back to main path loop

elif choice in ['b', 'check inventory']:

player.print\_inventory()

else:

print("Invalid option. Try again.\n".center(columns))

elif fight == 'b':

clear\_console()

print('You decide to retreat... Bok Bok!'.center(columns))

time.sleep(3)

return # Go back to main path loop

elif fight == 'c':

clear\_console()

print('You try to befriend the tree... and it attacks you!\n'.center(columns))

time.sleep(3)

player.take\_damage(20)

print(f'Your Health is now: {player.health}\n'.center(columns))

time.sleep(3)

print('The tree still stands there, rustling ominously...'.center(columns))

time.sleep(3)

elif fight == 'd':

clear\_console()

player.print\_inventory()

time.sleep(3)

else:

print("That’s not a valid option. Try again.".center(columns))

def forest(player, tree): # if youve already killed tree and come back

if tree.health == 0:

print('"You have already killed the beast that is here"\n'.center(columns))

time.sleep(3)

print('"Go Back. We have to save Helga"\n'.center(columns))

time.sleep(3)

print(' Feeling like an idiot, you turn around and go back'.center(columns))

time.sleep(3)

return

print('You venture into the forest, you hear whispers...\n'.center(columns)) # when you first enter forest / comeback aftr leaving but didnt kill

time.sleep(3)

print('“So, what makes you want to go through the forest?”\n'.center(columns))

centered\_input()

time.sleep(3)

print('\n')

print('“Uh huh . . . Okay . . . Whatever you say, Brochacho.”\n'.center(columns))

time.sleep(3)

print('\*bsh bhs bsh\*'.center(columns))

time.sleep(3)

tree\_picture()

tree\_battle(player, tree)

DOORS.PY

from settings import \*

import time

def trowser\_picture(): # end boss picture (Trowser)

clear\_console()

print(r'''

░░░░░░░░░░░░░░░░░░░░░░░░░░░░░

░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░

░ ░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░ ░ ░ ░ ░░░░ ░░░ ░░░░ ░░░░ ░░░ ░ ░ ░░░ ░░

░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░

░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░

░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░ ░ ░▒▒░░░░ ░░░░░░░░░░░░░░ ░░░░░░░░░░░░░

░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░▒░▒▓██████▓▓██▓█░ ░

░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░▒▓▒░░░░░░░▒████▓▓▓▓▓▓▓▓▓▓▓▓██▒▓░ ░░

░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░▓▒▒▓░░░░░░▒█▓▓█▓▒▓▓▓▓▒▓▓▓▓▓▓████ ░▓▓█░ ░

░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░▓▒░░█▒░░░░░▒▓▒▒▒▒▒▒▒▓▒▒▒▒▒▒▒▒▒▒▓▒░░▓░▒▓░░░░░░░░░░░░ ░░░░░░░░░░░░░░

░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░█▒░░▒█░░░░░▒▓▒▓▓▓▓▓▓▒▓▓▓▓▒▓▓▒▓▓████▒░░▓▒

░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░█▓░░░▒██▒▒▒▒▓█▓▓▓█████▓▓▓▓▓▓▓▓▓▓█▒░░░▒█▒ ░░

░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░▒█▒░░░░▒██████▓▓▓▓▓▓▓▓▓▓▓▓█████▓░░░░░▒▓░ ░ ░░ ░ ░░

░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░▒▓░░░▒█▓▒▒▒▒█▒▒▒▒▒▓▓▓▒▒▒▒▓▒▒▒▒▓█▒░░▒▓▒░░░░ ░░ ░░░ ░░░░░░░░░░░░░░░░░

░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░▒█▓▓▓█▓▓▓▓▓▒▓▓▓▓▓▓▓▓▓▓▓▓▓▓▒▓▓▓█▓▓▓█░░░░░ ░░░ ░░░░░░░░░

░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░▒▓▓██▓▒▓▓▓▓▓██▓▓▓▓██▓▓▓▓▓▓▓██▓█▒░░░░░░░░░░ ░░░░░░░░░░░░░░░

░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░█▓█▓▓█▒▒▓▒▒▒▓▒▒▓▓▓▒▒▓▓▒░▓▓▓▓▒█▒░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░

░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░██▓▓█ ▒▓███▓▓▓▓███▓░ ▓▓▓▓█░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░

░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░▒▓██▓▓▓▓▓▓▓▒░████████▒▒▓▓▓▓▓▓███▓▒░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░

░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░▒█▒░░░░░▒▓█████▒░░░░░░▒█████▓▒░░░░░▒█▒░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░

░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░▒▓▒░░░░░░░░░░░░░▓▒░░░░▒▓▒░░░░░░░░░░░░░▓▒░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░

░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░█▓░░░░░░░░░░░░░░▒░░░░░░░░░░░░░░░▒░░░░░▒█░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░

░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░█▓░░░░░▓▒░░░░░░░░░░░░░░░░░░░░░░░▒█░░░░▒█░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░

░░░░░░░░░░░░░░░░░░░ ░░░░░░░░░░▓▓░░░░███▓░░░░░░░░░░░░░░░░░░░░▒███▒░░░▒█░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░

░░░░░░░░░░░░░░░░░░░ ░░░░░░░░░░░▓▒░░░██▓▓░▓▒░░░░░▒██▒░░░░░▒▒▒▓▓▓█▒░░▒▓▒░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░

░░░ ░░ ░░ ░░░ ░░ ░░░░░░░░░░░░░░▒█▓░░░▒▓▓█▓▓░▒██▓▓▓▓▓▓██▓▒█▓█▓▓▒░░░▒█▒░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░

░░░ ░░ ░ ░░░░░░░░░░░░░░░▒██▒░▒█▓▓▓█░▓▓▓▓▓▓▓▓▓▓▒▒▓▓█▓░░░▓█▓░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░

░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░▒█░░░▓▒▒▓▓▒▒▒▒▒▓▓▒▓▓▓▒▓█▒░░░█▓░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░

░░░░░░░░░░░░░░ ░░ ░░░░░░░░░░░░░░░░░░▓█░░░▒▓▒▓░░▓▒▓▓▓▓░░▒▓█░░░▒▓▓░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░

░░░░ ░ ░ ░░░░░░░░░░░░░░░░▒█░░░▓█▒░▒██████▓░▒█░░░▒█▒░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░

░░░ ░░ ░░ ░░░░░░░░░░░░░░░░░░▒█▒░▒▓▒░░░░░░░░░░▒▓▒░██░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░

░░░░░░░░░░░░░░ ░░░░░░░░░░░░░░░░░▒█░░░░░░░░░░░░░░░░░░░▒█░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░

░ ░░░░░░░░░░░░░░▓▒░░░░░░░░░░░░░░░░░░░░█▓░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░

░░░▒█▓▒░░░▒▒▓█▓▓█▓▒▒░░░▒▓█░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░

░░▒▓▓▒▒░░░░░░░░▒▒▓▓▒░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░

░░░░░░░░░░░░░░ ░░░░░░░░░░░░░ ░░░░░░░░░ ░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░

░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░

░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░

░ ░░░░ ░░░ ░░ ░░░ ░░░ ░░ ░░░ ░ ░░░░░░░░░░░░░░░░░░░░░░░░░░░░░ ░░ ░░ ░░

░░░░ ░ ░░ ░░ ░ ░░ ░ ░ ░ ░░░░░░░░░░░░░░░░░░ ░░ ░ ░ ░

''')

time.sleep(5)

clear\_console()

def door\_picture(): # Door Ascii

clear\_console()

print(r"""

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

/ \ / |

/ Left | / Right |

+----------------------------+ +----------------------------+

| | | |

| | | |

| | | |

| | | |

| | | |

| (| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | (|

| | | This | | |

| | | is | | |

| | | a | | |

| | | sign | | |

| | |\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_| | |

| | | | |

| | | | |

+----------------------------+ | +----------------------------+

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

""")

time.sleep(5)

clear\_console()

def barry\_battle(player, barry): # Controls the first battle, the one with barry!

if barry.health == 0:

return True

while True:

print("You can either:".center(columns))

print("a) Fight Barry".center(columns))

print("b) Go back".center(columns))

print("c) Check Inventory".center(columns))

choice = centered\_input()

time.sleep(1)

if choice == 'a':

print("You attack Barry with your axe!".center(columns))

time.sleep(2)

print("barry throws his stick at you\n".center(columns))

time.sleep(2)

player.take\_damage(barry.damage)

barry.health -= player.weapon\_damage

if barry.health < 0:

barry.health = 0

print(f"{player.user}'s health: {player.health}".center(columns))

print(f"Barry's health: {barry.health}".center(columns))

time.sleep(2)

clear\_console()

if barry.health == 0:

print("YOU KILLED BARRY YOU MONSTER".center(columns))

for item in barry.drop\_items:

player.add\_to\_inventory(item)

return True

elif choice == 'b':

return False

elif choice == 'c':

player.print\_inventory()

else:

print("Invalid option.".center(columns))

clear\_console()

def harry\_battle(player, harry): # Second Battle

if harry.health == 0:

return True

while True:

print('You are now about to fight "harry"\n'.center(columns))

time.sleep(2)

print("You can either:".center(columns))

print("a) Fight Harry".center(columns))

print("b) Go back".center(columns))

print("c) Check Inventory".center(columns))

choice = centered\_input()

time.sleep(1)

if choice == 'a':

print('"harry harry harry" - you hear this guy chanting. trying to boost himself up'.center(columns))

time.sleep(2)

player.take\_damage(harry.damage)

harry.health -= player.weapon\_damage

if harry.health < 0:

harry.health = 0

print(f"{player.user}'s health: {player.health}".center(columns))

print(f"Harry's health: {harry.health}".center(columns))

time.sleep(2)

clear\_console()

if harry.health == 0:

print("Harry is defeated!".center(columns))

for item in harry.drop\_items:

player.add\_to\_inventory(item)

return True

elif choice == 'b':

return False

elif choice == 'c':

player.print\_inventory()

else:

print("Invalid option.".center(columns))

clear\_console()

def larry\_battle(player, larry): # 3rd

if larry.health == 0:

return True

while True:

print('"You See Larry. JEEZ 3 of em"\n'.center(columns))

time.sleep(2)

print("You can either:".center(columns))

print("a) Fight Larry".center(columns))

print("b) Go back".center(columns))

print("c) Check Inventory".center(columns))

choice3 = centered\_input()

time.sleep(1)

if choice3 == 'a':

print("You battle Larry, who swings wildly!".center(columns))

time.sleep(2)

player.take\_damage(larry.damage)

larry.health -= player.weapon\_damage

if larry.health < 0:

larry.health = 0

print(f"{player.user}'s health: {player.health}".center(columns))

print(f"Larry's health: {larry.health}".center(columns))

time.sleep(2)

clear\_console()

if larry.health == 0:

print("Larry collapses, defeated.".center(columns))

for item in larry.drop\_items:

player.add\_to\_inventory(item)

return True

elif choice3 == 'b':

return False

elif choice3 == 'c':

player.print\_inventory()

else:

print("Invalid option.".center(columns))

clear\_console()

def garry\_battle(player, garry): # 4th

if garry.health == 0:

return True

while True:

print('"Oh my god! Where is Trowser . . ."\n'.center(columns))

time.sleep(2)

print("You can either:".center(columns))

print("a) Fight Garry".center(columns))

print("b) Go back".center(columns))

print("c) Check Inventory".center(columns))

choice4 = centered\_input()

time.sleep(1)

if choice4 == 'a':

print("You lunge at Garry for a showdown!".center(columns))

time.sleep(2)

player.take\_damage(garry.damage)

garry.health -= player.weapon\_damage

if garry.health < 0:

garry.health = 0

print(f"{player.user}'s health: {player.health}".center(columns))

print(f"Garry's health: {garry.health}".center(columns))

time.sleep(2)

clear\_console()

if garry.health == 0:

print("Garry is vanquished. one enemy to go,.".center(columns))

for item in garry.drop\_items:

player.add\_to\_inventory(item)

return True

elif choice4 == 'b':

return False

elif choice4 == 'c':

player.print\_inventory()

else:

print("Invalid option.".center(columns))

clear\_console()

def trowser\_battle(player, trowser): # Controls the last battle

if trowser.health == 0:

return True

trowser\_picture()

clear\_console()

while True:

print('"You Have made it. You Are now able to save Helga"\n'.center(columns))

time.sleep(3)

print("You can either:".center(columns))

print("a) Fight trowser".center(columns))

print("b) Go back".center(columns))

print("c) Check Inventory".center(columns))

choice5 = centered\_input()

time.sleep(1)

if choice5 == 'a':

print("You attack trowser with your axe!".center(columns))

time.sleep(2)

print("Trowsers jumps at you\n".center(columns))

time.sleep(2)

player.take\_damage(trowser.damage)

trowser.health -= player.weapon\_damage

if trowser.health < 0:

trowser.health = 0

print(f"{player.user}'s health: {player.health}".center(columns))

print(f"trowser's health: {trowser.health}".center(columns))

time.sleep(2)

clear\_console()

if trowser.health == 0:

print("YOU KILLED TROWSER!!!".center(columns))

time.sleep(3)

print('"H- Hello is that you Dime ??" - Helga'.center(columns))

print('"HELGAAAAAAAAAA"'.center(columns))

time.sleep(2)

clear\_console()

print('This was -'.center(columns))

time.sleep(3)

clear\_console()

print(r'''

████████╗██╗ ██╗███████╗

╚══██╔══╝██║ ██║██╔════╝

██║ ███████║█████╗

██║ ██╔══██║██╔══╝

██║ ██║ ██║███████╗

╚═╝ ╚═╝ ╚═╝╚══════╝

██╗ ███████╗ ██████╗ ███████╗███╗ ██╗██████╗

██║ ██╔════╝██╔════╝ ██╔════╝████╗ ██║██╔══██╗

██║ █████╗ ██║ ███╗█████╗ ██╔██╗ ██║██║ ██║

██║ ██╔══╝ ██║ ██║██╔══╝ ██║╚██╗██║██║ ██║

███████╗███████╗╚██████╔╝███████╗██║ ╚████║██████╔╝

╚══════╝╚══════╝ ╚═════╝ ╚══════╝╚═╝ ╚═══╝╚═════╝

██████╗ ███████╗

██╔═══██╗██╔════╝

██║ ██║█████╗

██║ ██║██╔══╝

╚██████╔╝██╗

╚═════╝ ╚═╝

██╗ ██╗███████╗██╗ █████╗ █████╗

██║ ██║██╔════╝██║ ██╔═══╝ ██╔══██╗

███████║█████╗ ██║ ██║ ███╗ ███████║

██╔══██║██╔══╝ ██║ ██║ ██║ ██╔══██║

██║ ██║███████╗███████╗╚██████╔╝ ██║ ██║

╚═╝ ╚═╝╚══════╝╚══════╝ ╚═════╝ ╚═╝ ╚═╝

''')

time.sleep(5)

clear\_console()

print('Congrats on Beating "The Legand of Helga". I hope you enjoyed it <3'.center(columns))

time.sleep(5)

exit()

elif choice5 == 'b':

return False

elif choice5 == 'c':

player.print\_inventory()

else:

print("Invalid option.".center(columns))

clear\_console()

def field\_of\_foe(player, dime, barry, harry, larry, garry, trowser): # where all is collated and to be called in choice.py

from choice import choose\_path

if not hasattr(player, 'entered\_field\_of\_foe'): # to check what battle you are up to

player.entered\_field\_of\_foe = False

if not player.entered\_field\_of\_foe:

print("You and Dime enter the Field of Foe...".center(columns))

time.sleep(3)

door\_picture()

print("You read the mysterious sign...".center(columns))

time.sleep(3)

while True:

print("What door do you open?".center(columns))

print("a) The left door".center(columns))

print("b) The right door".center(columns))

door = centered\_input()

if door == 'a':

print('The left door contains a battle!'.center(columns))

time.sleep(2)

clear\_console()

break

elif door == 'b':

print("You open the right door and see no enemy.".center(columns))

time.sleep(2)

print("You then glance left and realize...".center(columns))

time.sleep(2)

print("These doors aren't actually separated.".center(columns))

time.sleep(2)

print("A small man named Barry is waiting for you regardless.".center(columns))

time.sleep(3)

clear\_console()

break

else:

print("Invalid choice. Pick left or right.".center(columns))

clear\_console()

# Mark that intro was completed

player.entered\_field\_of\_foe = True

# Now proceed to battles based on progress

while player.field\_of\_foe\_stage < 5:

if player.field\_of\_foe\_stage == 0:

success = barry\_battle(player, barry)

if success:

player.field\_of\_foe\_stage += 1

print("A path opens forward...".center(columns))

time.sleep(2)

else:

choose\_path(player, dime, tree, barry, harry, larry, garry, trowser)

return

elif player.field\_of\_foe\_stage == 1:

success = harry\_battle(player, harry)

if success:

player.field\_of\_foe\_stage += 1

print("Another foe awaits...".center(columns))

time.sleep(2)

else:

choose\_path(player, dime, tree, barry, harry, larry, garry, trowser)

return

elif player.field\_of\_foe\_stage == 2:

success = larry\_battle(player, larry)

if success:

player.field\_of\_foe\_stage += 1

print("Almost there... few remains.".center(columns))

time.sleep(2)

else:

choose\_path(player, dime, tree, barry, harry, larry, garry, trowser)

return

elif player.field\_of\_foe\_stage == 3:

success = larry\_battle(player, garry)

if success:

player.field\_of\_foe\_stage += 1

print("You’ve defeated them all! Atleast, all but one!.".center(columns))

time.sleep(2)

else:

choose\_path(player, dime, tree, barry, harry, larry, garry, trowser)

return

elif player.field\_of\_foe\_stage == 4:

success = trowser\_battle(player, trowser)

if success:

player.field\_of\_foe\_stage += 1

print("You have defeated Trowser! Congratualations, You have completed the game.".center(columns))

time.sleep(2)

return

else:

choose\_path(player, dime, tree, barry, harry, larry, garry, trowser)

return

BAR.PY

from settings import \*

def swoosh(): # cowboy door for tavern. This functions holds the ascii art of said door

clear\_console()

swoosh\_ascii = '''

=========#%%%%\* :#%%%%#+========

=========#%%##\* :#%%%%#=========

=========#%%%%\* :==+++++#++\*\*\*\*=. :%%%%%#=========

=========\*%%%%\*\*\*\*\*+=======+++++++++++\*#\*\*\*\*\*\*\*\*\*\*\*+===---=\*\*\*#\*=%%%%%#+========

=========\*%%%%\*\*+\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*=%%%%%#=========

-=--=====#%%%%\*\*+\*\*\*%%%%%%%%%%%%%%%%\*\*\*#\*\*\*#%%%%%%%%%%%%%%%\*\*\*=+=%%%%%#=========

=-=======\*%%%%#\*+\*\*\*%%%%%%%%%%%%%%%%\*\*\*#\*\*\*#%%%%%%%%%%%%%%%#\*\*++=%%%%%\*=========

-=--=====#%%%%#\*\*\*\*\*%%%%%%%%%%%%%%%%\*\*\*%\*\*\*\*%%%%%%%%%%%%%%%###%\*=%%%%%#=========

---=-====#%%%%#\*\*\*\*\*%%%%%%%%%%%%%%%%\*\*\*#\*\*\*#%%%%%%%%%%%%%%%\*\*#%#+%%%%%\*=========

--=-=====\*%%%%#\*\*\*\*\*%%%%%%%%%%%%%%%%\*\*\*#\*\*\*#%%%%%%%%%%%%%%%###%#+%%%%%#=========

-----====#%%%%#\*\*\*\*\*\*\*\*#############\*\*\*%\*\*\*\*\*\*\*\*\*#############%%+%%%%%\*=========

-----====#%%%%#\*\*\*\*\*\*\*##############\*+\*%\*\*\*\*\*\*##############%%%%+%%%%%\*=========

-----====\*%%%%#\*\*\*\*\*\*\*\*\*\*\*\*\*\*#\*\*\*\*\*\*\*\*\*%\*\*\*\*\*\*\*\*\*\*\*\*\*#\*\*\*\*\*###%%+%%%%%\*=========

------===#%%%%#\*\*\*\*\*\*\*\*##\*##########\*\*\*%\*\*\*\*#\*################%%+%%%%%\*=========

-----=-==\*%%%%#\*\*\*\*\*\*##############\*\*\*\*%\*\*\*\*##################%%+%%%%%\*=======-=

-----====\*%%%%#\*\*\*\*#%%%%%%%%%%%%%%%%\*\*\*%\*\*\*#%%%%%%%%%%%%%%%%%%%%+%%%%%\*========-

------===\*%%%%#\*\*\*\*#%%%%%%%%%%%%%%%%\*\*\*%\*\*\*#%%%%%%%%%%%%%%%%%%%%+%%%%%\*========-

------===\*%%%%%\*\*\*\*#%%%%%%%%%%%%%%%%\*\*\*%##\*#%%%%%%%%%%%%%%%##%%%+%%%%%\*====-=-=-

-------==\*%%%%#\*#\*##%%%%%%%%%%%%%%%%#\*#%####%%%%%%%%%%%%%%%%#%%%+%%%%%\*=====----

------===\*%%%%%\*\*\*\*#\*\*\*\*\*\*\*\*########\*##%####\*\*\*\*\*\*\*#########\*#+#+%%%%%\*===--=---

-------==\*%%%%%\*\*\*#################\*###%####################\*#\*#+%%%%%\*===------

:-:----==\*%%%%%\*#\*\*#\*################\*#%###########\*\*\*\*\*\*\*\*##%%%+%%%%%\*==-------

::-----==\*%%%%%\*\*\*\*\*\*\*\*\*\*\*=-:. ::::::: ..:=\*\*\*\*\*##%%%%+%%%%%\*===------

:::----==\*%%%%%+++\*=-:: .:-=\*\*\*\*=%%%%%\*==------:

::::---==\*%%%%# :%%%%%#==---:-::

:-:----==\*%%%%# Like a bar door used in cowboy movies:p :%%%%%#==---::::

'''

swoosh\_lines = swoosh\_ascii.strip('\n').split('\n')

vertical\_padding = (rows - len(swoosh\_lines)) // 2

print('\n' \* vertical\_padding)

for line in swoosh\_lines:

print(line.center(columns))

time.sleep(5)

clear\_console()

def enter\_bar(player, dime, tree): # Holds output of first entering bar and checks to make sure its you first, if 2nd or more, prints other stuff

from choice import choose\_path

clear\_console()

global enter

swoosh()

if enter == 1: # If you've already entered.

print('\*swish swoosh\*\n'.center(columns))

time.sleep(3)

print('"You Again? I Already told you, Trowser is in the Field of Foe" - Cad Bane\n'.center(columns))

time.sleep(3)

print('"Dude we have already been here, why did you come back"\n'.center(columns))

time.sleep(3)

print(' You exit the bar'.center(columns))

time.sleep(3)

return choose\_path(player, dime, tree, barry, harry, larry, garry, trowser)

enter += 1

print('\*swish swoosh\*\n'.center(columns))

time.sleep(3)

print('You step into the tavern, the smell of ale and stories thick in the air...\n'.center(columns))

time.sleep(3)

print('\*You hear\* \n'.center(columns))

time.sleep(1)

print('"NO WEAPONS"\n'.center(columns))

time.sleep(1)

print('Yelled across the bar from the Bartender\n'.center(columns))

time.sleep(1)

print('You Put your weapons at the door and continue walking in with Dime. \n'.center(columns))

time.sleep(3)

print('You sit down at the bar and order a drink\n'.center(columns))

time.sleep(3)

print('"Eyy What are you here for? - stranger"\n'.center(columns))

time.sleep(3)

print('"something you need? You arent our usual folk" - stranger\n'.center(columns))

time.sleep(3)

print('I uhh . . . We are . . .\n '.center(columns))

time.sleep(3)

print('"Someone we know was kidnapped. we need infomation."\n'.center(columns))

time.sleep(3)

print('Yeah, what he said\n'.center(columns))

time.sleep(3)

print('\*The stranger chewing on a tooth pick\*\n.'.center(columns))

time.sleep(3)

print('"I Think i can be of service" - stranger \n'.center(columns))

time.sleep(3)

print('\*Tips his cowboy hat down to you\*\n'.center(columns))

time.sleep(3)

print('"First im going to need some names." - stranger\n'.center(columns))

time.sleep(2.5)

print(f'"Im Dime and he is {player.user} we are looking some by the name Helga"\n'.center(columns))

time.sleep(3)

print('\*Still chewing tooth pick but moves it to other side\*\n'.center(columns))

time.sleep(3)

print('"That rings a bell, Helga. I might be able to help" - stranger\n'.center(columns))

time.sleep(3)

print('Thanks, We never got you name.\n'.center(columns))

time.sleep(3)

print('"Bane, Cad Bane" - stranger\n'.center(columns))

time.sleep(3)

print('"Go sit in that booth there and wait for me" - Cad Bane\n'.center(columns)) # totally not a taken name

time.sleep(3)

print('\*You and Dime walk to the booth at which Bane pointed at and wait\*\n'.center(columns))

time.sleep(3)

clear\_console()

print('"Who is this Helga to you" - Cad Bane\n'.center(columns))

question(player, dime, tree)

def question(player, dime, tree): # THe questions that are asked

while True:

print('a) I dont know her, Dimes friend'.center(columns))

print('b) Our Friend'.center(columns))

print('c) My Mother'.center(columns))

print('d) Thats not of your concern'.center(columns))\

stranger = centered\_input()

time.sleep(3)

clear\_console()

if stranger == 'a': # if answer == so and so, do this function

pet(player, dime, tree)

elif stranger == 'b':

friend(player, dime, tree)

elif stranger == 'c':

mother(player, dime, tree)

elif stranger == 'd':

grumpy(player, dime, tree)

else:

print('Invalid input. Try again.'.center(columns))

def pet(player, dime, tree): # question 1

print('What are you some pet?" - Cad Bane'.center(columns))

print('a) Yes'.center(columns))

print('b) No'.center(columns))

pet = centered\_input()

print('\n')

time.sleep(3)

if pet == 'b':

print('"Whatever you say. - Cad Ban\n'.center(columns))

time.sleep(3)

trowser(player, dime, tree)

elif pet == 'a':

print('"Uhhhhh . . . ." - Cad Bane\n'.center(columns))

time.sleep(3)

print('Will you still help us?\n'.center(columns))

time.sleep(3)

print('"I Guess" - Cad Bane\n'.center(columns))

time.sleep(3)

trowser(player, dime, tree)

else:

print('"Weird way to answer a yes or no question" - Cad Bane\n'.center(columns))

time.sleep(3)

trowser(player, dime, tree)

def friend(player, dime, tree): # question 2

print('"I Doubt she is friends with YOU." - Cad Bane\n'.center(columns))

time.sleep(3)

trowser(player, dime, tree)

def mother(player, dime, tree): # question 3

print('"Oh Really? Shes YOUR mother? She had no kids idiot\n" - Cad Bane'.center(columns))

time.sleep(3)

trowser(player, dime, tree)

def grumpy(player, dime, tree): # question 4

print('"Oh Mr Grumpy pants over here." - Cad Bane\n'.center(columns))

time.sleep(3)

trowser(player, dime, tree)

def trowser(player, dime, tree): # where the boss is locatated

from choice import choose\_path

clear\_console()

print('"Helga was taken by a Humaniod Turtle called Trowser" - Cad Bane'.center(columns))

time.sleep(3)

print('a) Trowser?'.center(columns))

print('b) Ive heard of him. Where is he?'.center(columns))

whereabouts = centered\_input()

time.sleep(3)

print('\n')

if whereabouts == 'a':

print('"The Humanoid Turtle? I just said that, anyways, go through the Field of Foe" - Cad Bane\n'.center(columns))

time.sleep(3)

return choose\_path(player, dime, tree, barry, harry, larry, garry, trowser)

elif whereabouts == 'b':

print('"I doubt you of all people have heard of him. He can be found in the Field of Foe" - Cad Bane\n'.center(columns))

time.sleep(3)

return choose\_path(player, dime, tree, barry, harry, larry, garry, trowser)

else:

print('"Weird way to answer a yes or no question - Cad Bane\n"'.center(columns))

time.sleep(3)

return trowser(player, dime, tree)

SETTINGS.PY

import sys

import time

import shutil

import os

def clear\_console(): # clears the terminal

os.system('cls' if os.name == 'nt' else 'clear')

# Abstraction (1/4)

def centered\_input(prompt='>> '): # centers input statements

return input(' ' \* ((columns - len(prompt)) // 2) + prompt).lower() # centers inoput statements

enter = 0

columns, rows = shutil.get\_terminal\_size() # gets terminal size for centering

# ALl Class are encapsulation (2/4)

class Race: # race class

def \_\_init\_\_(self, name, strength, health\_bonus):

*self*.name = name

*self*.strength = strength

*self*.health\_bonus = health\_bonus

human = Race(

"Human",

strength=5,

health\_bonus=0

)

elf = Race(

"Elf",

strength=3,

health\_bonus=-10

)

orc = Race(

"Orc",

strength=9,

health\_bonus=20

)

dwarf = Race(

"Dwarf",

strength=7,

health\_bonus=30

)

worm = Race(

"Worm",

strength=6,

health\_bonus=-5

)

grug = Race( # SECRET RACES

"Grug",

strength=40,

health\_bonus=200

)

no = Race(

"No",

strength=0,

health\_bonus=-95

)

races = { # Dictionary

"human": human,

"elf": elf,

"orc": orc,

"dwarf": dwarf,

"worm": worm,

"grug": grug,

"no": no,

}

class Weapon: # Weapon controls, such as damage, etc

def \_\_init\_\_(self, name, base\_damage):

*self*.name = name

*self*.base\_damage = base\_damage

axe = Weapon(

"Axe",

base\_damage=8

)

dimes\_sword = Weapon(

"Dime's Sword",

base\_damage=10

)

class Character: # everything to do with players, inventory, enemies, dime, etc

def \_\_init\_\_(self, name, health, race=None, weapon=None):

*self*.user = name

*self*.race = race

*self*.max\_health = health

*self*.health = health

*self*.weapon = weapon

*self*.weapon\_damage = (weapon.base\_damage + race.strength) if weapon and race else 0

*self*.defense = 1.0

*self*.inventory = []

def take\_damage(self, amount):

# Polymorphism. If player has 0 health. prints you die message, if others. prints so and so died. (3/4)

actual\_damage = max(0, int(amount \* *self*.defense)) # apply defense multiplier

*self*.health -= actual\_damage

if *self*.health < 0:

*self*.health = 0

if *self*.health == 0:

print(f"{*self*.user} has died.".center(columns))

if isinstance(*self*, Player):

print("You died, idiot. Game over.".center(columns))

sys.exit()

return actual\_damage

class Player(Character): # Player Class

# Inheritance, same as Dime and Enemie classes (4/4)

def \_\_init\_\_(self, name, race):

super().\_\_init\_\_(name, 100 + race.health\_bonus, race, axe)

*self*.weapon\_damage = *self*.weapon.base\_damage + race.strength

*self*.inventory = [axe\_item]

*self*.equipped\_equipment = None

*self*.equipped = {

"head": None,

"feet": None,

"chest": None,

"pants": None,

"shield": None

}

*self*.field\_of\_foe\_stage = 0

def add\_to\_inventory(self, item): # adding items to ibv

*self*.inventory.append(item)

print(f"{item.name} added to inventory.".center(columns))

def use\_item(self, using): # using items

for item in *self*.inventory:

if item.name.lower() == using.lower():

if item.use\_effect:

item.use\_effect(*self*)

if item.type == "consumable":

*self*.inventory.remove(item)

return

else:

print(f"{item.name} can't be used.".center(columns))

return

print("Item not found in inventory.".center(columns))

def equip\_item(self, using): # equiping items

for item in *self*.inventory:

if item.name.lower() == using.lower():

if item.type == "equipment" and item.equip\_effect and item.slot:

*self*.equipped[item.slot] = item

item.equip\_effect(*self*)

return

elif item.type == "weapon":

if using.lower() in weapon\_registry:

*self*.weapon = weapon\_registry[using.lower()]

*self*.weapon\_damage = *self*.weapon.base\_damage + *self*.race.strength

print(f"{item.name} equipped as weapon. Damage: {*self*.weapon\_damage}".center(columns))

return

else:

print(f"{item.name} is not a valid weapon object.".center(columns))

return

else:

print(f"{item.name} can't be equipped.".center(columns))

return

print("Item not found in inventory.".center(columns))

def print\_inventory(self):

while True:

clear\_console()

print(f"{'No.':<4} {'Name':<25} {'Type':<12} {'Description'}".center(columns))

print(('-' \* 70).center(columns))

for idx, item in enumerate(*self*.inventory, 1):

# Mark equipped items

if item.type == 'weapon' and *self*.weapon and *self*.weapon.name == item.name:

display\_name = f"[{item.name}]"

elif item.type == 'equipment' and item in *self*.equipped.values():

display\_name = f"[{item.name}]"

elif item.type == 'equipment':

display\_name = item.name

else:

display\_name = item.name

print(f"{idx:<4} {display\_name:<25} {item.type:<12} {item.description}".center(columns))

print(('-' \* 70).center(columns))

print('Choose an action: EQUIP (A), USE (B), BACK (C): '.center(columns))

selection = centered\_input()

if selection == 'c':

clear\_console()

break # exit inventory

elif selection in ['a', 'b']:

try:

print("Enter the item number:".center(columns))

item\_number = centered\_input()

item\_number = int(item\_number)

if 1 <= item\_number <= len(*self*.inventory):

item = *self*.inventory[item\_number - 1]

else:

print("Invalid item number.".center(columns))

continue

except ValueError:

print("Please enter a valid number.".center(columns))

continue

if selection == 'a': # Equip

if item.type in ['weapon', 'equipment']:

*self*.equip\_item(item.name)

else:

print(f"{item.name} cannot be equipped.".center(columns))

elif selection == 'b': # Use

if item.type == 'consumable':

*self*.use\_item(item.name)

else:

print(f"{item.name} cannot be used.".center(columns))

else:

print("Invalid selection. Please choose A, B, or C.".center(columns))

class Dime(Character): # dime fucntion

def \_\_init\_\_(self):

super().\_\_init\_\_("Dime", 100, human, dimes\_sword)

*self*.weapon\_damage = *self*.weapon.base\_damage + *self*.race.strength

class Enemy(Character): # controls enemies

def \_\_init\_\_(self, name, health, damage, drop\_items=None):

super().\_\_init\_\_(name, health)

*self*.damage = damage

*self*.drop\_items = drop\_items or []

class Item: # Cotnrols items

def \_\_init\_\_(self, name, use\_effect=None, equip\_effect=None, type='consumable', description='No Description', slot=None):

*self*.name = name

*self*.use\_effect = use\_effect

*self*.equip\_effect = equip\_effect

*self*.type = type

*self*.description = description

*self*.slot = slot

axe\_item = Item( # items stats

"Axe",

type='weapon',

description='Your axe used to chop wood.'

)

dimes\_sword\_item = Item(

"Dime's Sword",

type='weapon',

description="Dimes sword he tried to attack you with"

)

weapon\_registry = {

"axe": axe,

"dime's sword": dimes\_sword,

}

def use\_sap\_of\_life(player): # item and equipment controlling (for those below aswell)

heal\_amount = 30

player.health += heal\_amount

if player.health > player.max\_health:

player.health = player.max\_health

print(f"{player.user} used the Sap of Life and healed {heal\_amount} health! Current health: {player.health}".center(columns))

time.sleep(3)

def use\_barrys\_tears(player):

heal\_amount = 10

player.health += heal\_amount

if player.health > player.max\_health:

player.health = player.max\_health

print(f"{player.user} drank barrys tears like a wierdo but healed {heal\_amount} health! Current health: {player.health}".center(columns))

time.sleep(3)

def equip\_bark\_shield(player):

player.defense = 0.2

print(f"{player.user} equipped the Bark Shield. Less damage will be taken!".center(columns))

def equip\_bucket\_hat(player):

player.defense = 0.2

print(f"{player.user} put on the stolen hat and now takes a little less damage".center(columns))

def equip\_sandals(player):

player.defense = 0.2

print(f"{player.user} put on some fresh kicks!".center(columns))

def equip\_brown\_stained\_pants(player):

player.defense = 0.2

print(f"Why would you want this equiped".center(columns))

def equip\_chopping\_board\_chest\_plate(player):

player.defense = 0.2

print(f"{player.user} put on some a wooden board".center(columns))

sap\_of\_life = Item( # stats once again (fo below aswell)

"Sap of Life",

use\_effect=use\_sap\_of\_life,

type='consumable',

description='Mystical tree juice, capable of healing'

)

barrys\_tears = Item(

"Tears of Barry",

use\_effect=use\_barrys\_tears,

type='consumable',

description='Tears from the unstoppable Barry'

)

bark\_shield = Item(

"Bark Shield",

equip\_effect=equip\_bark\_shield,

type='equipment',

description='Bark Ripped off the tree, deflects damage.',

slot='shield'

)

bucket\_hat = Item(

"Bucket Hat",

equip\_effect=equip\_bucket\_hat,

type='equipment',

description='You Stole his hat? Meany',

slot='head'

)

sandals = Item(

"Sandals",

equip\_effect=equip\_sandals,

type='equipment',

description='you robbed him for his shoes?',

slot='feet'

)

brown\_stained\_pants = Item(

"Brown Stained Pants",

equip\_effect=equip\_brown\_stained\_pants,

type='equipment',

description='Some Pants with an awful smell . . . ',

slot='pants'

)

chopping\_board\_chest\_plate = Item(

"Chopping Board Chest Plate",

equip\_effect=equip\_chopping\_board\_chest\_plate,

type='equipment',

description='With a chopping board and tape you are protected from danger',

slot='chest'

)

tree = Enemy(

"Tree",

health=200,

damage=10,

drop\_items=[bark\_shield,

sap\_of\_life]

)

barry = Enemy(

"Barry",

health=15,

damage=2,

drop\_items=[barrys\_tears]

)

harry = Enemy(

"Harry",

health=30,

damage=4,

drop\_items=[sandals, bucket\_hat]

)

larry = Enemy(

"Larry",

health=45,

damage=6,

drop\_items=[brown\_stained\_pants]

)

garry = Enemy(

"Garry",

health=60,

damage=10,

drop\_items=[chopping\_board\_chest\_plate]

)

trowser = Enemy(

"Trowser",

health=200,

damage=20,

)